repository.ub.ac.id

IMPLEMENTASI METODE *ARTIFICIAL NEURAL NETWORK (ANN) MODEL BASED ON USE CASE POINT* DALAM MENGHITUNG BIAYA PERANGKAT LUNAK (STUDI KASUS CV. PROFILE IMAGE STUDIO)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ade Pratama
NIM: 145150400111044

PROGRAM STUDI SISTEM INFORMASI
JURUSAN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

PENGESAHAN

IMPLEMENTASI METODE *ARTIFICIAL NEURAL NETWORK (ANN) MODEL BASED ON USE CASE POINT* DALAM MENGHITUNG BIAYA PERANGKAT LUNAK
(STUDI KASUS CV. PROFILE IMAGE STUDIO)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ade Pratama
NIM: 145150400111044

Skripsi ini telah diuji dan dinyatakan lulus pada
27 Desember 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Andi Reza Perdanakusuma, S.Kom., M.MT.	Djoko Pramono, S.T., M.Kom.
NIK. 201607 861128 1 001	NIP. 19780108 200501 1 002

Mengetahui
Ketua jurusan Sistem Informasi

Herman Tolle, Dr. Eng., S.T, M.T
NIP: 19740823 200012 1 001

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 27 Desember 2018

Ade Pratama
NIM: 145150400111044

UNIVERSITAS BRAWIJAYA

KATA PENGANTAR

Puji dan syukur penulis ucapkan kepada Tuhan Yesus Kristus yang memberikan penulis berkat dan kekuatan dalam menyelesaikan skripsi ini. Skripsi ini dapat terselesaikan tidak lepas dari dukungan dari banyak pihak yang telah menolong penulis. Oleh karena itu, pada kesempatan ini penulis hendak mengucapkan terimakasih kepada:

1. Bapak Andi Reza Perdanakusuma, S.Kom., M.MT, selaku Dosen Pembimbing I yang telah memberikan banyak waktu bimbingan maupun masukan selama pengerjaan skripsi agar penulis dapat menyelesaikan skripsi ini dengan sebaik-baiknya.
2. Bapak Djoko Pramono, S.kom., M.Eng selaku Dosen Pembimbing II yang telah memberikan ide, masukan, serta waktu dan perhatiannya supaya penulis dapat menyelesaikan skripsi ini dengan sebaik-baiknya.
3. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D Firdaus selaku dekan Fakultas Ilmu Komputer Universitas Brawijaya.
4. Bapak Herman Tolle, Dr. Eng., S.T, M.T selaku ketua jurusan sistem informasi Universitas Brawijaya.
5. Bapak Yusi Tyroni Yusi Tyroni Mursityo, S.Kom., M.AB. selaku ketua program studi sistem informasi Universitas Brawijaya.
6. Bapak Andi Reza Perdanakusuma, S.Kom., M.MT, selaku Dosen Pembimbing I yang telah memberikan banyak waktu bimbingan maupun masukan selama pengerjaan skripsi agar penulis dapat menyelesaikan skripsi ini dengan sebaik-baiknya.
7. Orangtua dan keluarga tercinta penulis yang selalu memberikan doa serta motivasi.
8. Pihak CV. Profile Image Studio yang telah membantu mendukung penulis menyelsaikan skripsi ini.

Demikianlah yang penulis dapat sampaikan. Dia akhir kata penulis mengucap syukur atas selesainya skripsi ini, dan penulis berterimakasih kepada semua pihak yang telah menolong penulis, baik dosen pembimbing dan sahabat yang tidak dapat penulis sebutkan satu per satu. Semoga Tuhan memberkati kita semua.

Malang, 27 Desember 2018

Penulis

adepratamaa99@gmail.com

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRAK

Estimasi biaya perangkat lunak merupakan aspek yang sangat penting dalam proyek teknologi informasi untuk pembuatan anggaran. Dalam proses penentuan harga suatu sistem CV. Profile Image Studio tidak menggunakan format standar ilmiah, mereka hanya menggunakan metode *guesstimate*. Metode tersebut hanya berdasarkan perkiraan yang dilandaskan pengalaman sebelumnya tanpa adanya bukti. Berdasarkan hal tersebut maka dilakukan implementasi metode lain dalam menghitung biaya perangkat lunak sebagai masukan kepada CV. Profile Image Studio. Estimasi biaya pada penelitian ini akan mengimplementasikan metode *Artificial Neural Network (ANN) model Based On Use Case Point.* Estimasi biaya dapat diperoleh setelah mendapatkan nilai estimasi effort atau usaha, sumber daya manusia serta estimasi waktu pengembangan sistem. Metode *Artificial Neural Network (ANN) model Based On Use Case Point* diimplementasikan kepada 2 sistem yang telah selesai dikembangkan oleh CV. Profile Image Studio. Pada akhir penelitian ini, dilakukan perbandingan biaya pengembangan sistem, dengan membandingkan hasil estimasi biaya yang diperoleh menggunakan metode *Artificial Neural Network (ANN) model Based On Use Case Point* dengan alokasi biaya yang dikeluarkan oleh CV. Profile Image Studio. Hasil implementasi metode *Artificial Neural Network (ANN) model Based On Use Case Point* dalam menghitung biaya perangkat lunak didapatkan bahwa total estimasi biaya sistem DBA ticketing sebesar Rp 17.703.140 yang dikerjakan oleh 12 orang dalam waktu 1247,46 jam sedangkan untuk sistem pintu air sebesar Rp 6.436.860 yang dikerjakan oleh 12 orang dalam waktu 453,6 jam.

Kata kunci: *estimasi biaya, estimasi biaya perangkat lunak, use case point, jaringan saraf tiruan*

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

ABSTRACT

Software cost estimation is an important part of an information technology project for budgeting. In the process of determining the price of a system CV. The Profile Image Studio does not use the scientific standard format or is commonly called parametric, they only use the guesstimate method. It is only estimates based on intuition. Based on this case, the cost estimation in this study will implement Artificial Neural Network (ANN) model Based on Use Case Point method is implemented in calculating software costs as suggestion for CV. Profile Image Studio. Cost estimates can be obtained after obtaining estimated effort, number of human resources and estimated system development time. Artificial Neural Networks (ANN) Based on Use Case Point method is implemented in 2 systems that have been developed by CV. Studio Profile Image. At the end of this study, the cost comparison of system development was done by comparing estimated cost results obtained using the Artificial Neural Network (ANN) model based on the Use Case Point with the allocation of costs incurred by CV. Studio Profile Image. The results of applying the Artificial Neural Network (ANN) model Based on the Use Case Point method in calculating software costs, it was found that the total estimated cost of the DBA ticket system was Rp. 17,703,140 carried out by 12 people in 1247.46 hours while the pintu air system was Rp. 6,436,860 carried out by 12 people in 453.6 hours.

Keyword: cost estimation, software cost estimation, use case point, artificial neural network

UNIVERSITAS BRAWIJAYA

DAFTAR ISI

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

DAFTAR TABEL

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

DAFTAR GAMBAR

UNIVERSITAS BRAWIJAYA

DAFTAR LAMPIRAN

repository.ub.ac.id

UNIVERSITAS BRAWIJAYA

repository.ub.ac.id

BAB 1 PENDAHULUAN

1.1 Latar belakang

Manajemen proyek TI berarti menerapkan pengetahuan, keterampilan, alat, dan teknik guna merancang kegiatan agar memenuhi harapan dan kebutuhan pihak *stakeholder* dari sebuah proyek teknologi informasi (PMI, 2000). Yang mana 10 bidang pengetahuan tersebut adalah manajemen integrasi, ruang lingkup, waktu, biaya, kualitas, sumber daya manusia, komunikasi, manajemen risiko, pengadaan, dan pemangku kepentingan (Schwalbe, 2014). Menerapkan manajemen proyek bagi perusahaan, pemerintahan dan organisasi untuk sebuah proyek IT adalah hal penting (Schwalbe, 2014). Ada perusahaan yang tidak menerapkan manajemen proyek dengan baik dan hal tersebut yang sering menjadi masalah dalam pengembangan suatu proyek teknologi informasi (Maswinandar, 2016). Sebagai contohnya pada bulan November 2012 angkatan udara Amerika Serikat akhirnya menghentikan proyek ERP yang mereka kembangkan setelah biayanya melampaui US$ 1 Milyar (Kanaracus, 2012). Menurut (Kashyap, 2014), dalam membuat anggaran, pengendalian dan perencanaan menghitung estimasi biaya perangkat lunak bersifat sangat penting. Sebuah perencanaan yang baik dibutuhkan dalam mengalokasikan biaya demi mengembangkan perangkat lunak dengan waktu dan biaya yang terbatas.

Estimasi biaya perangkat lunak merupakan proses memprediksi *effort* yang diperlukan untuk mengembangkan sebuah perangkat lunak. Estimasi biaya perangkat lunak meliputi dari 1) *Effort*; 2) *Project Duration*; 3) *Cost* (Leung et al., 2002). Kegagalan proyek bisa disebabkan karena scope, waktu, biaya, kualitas, sumber daya manusia, komunikasi, risiko, serta perubahan lingkungan (PMBOK, 2013). Ketika suatu proyek gagal maka akan menyebabkan kerugian bagi perusahaan itu sendiri, baik secara investasi, produktivitas dan reputasi.

Di CV. Profile Image Studio sendiri dalam melakukan estimasi biaya pembuatan perangkat lunak dengan cara melihat dari tingkat kompleksitas suatu perangkat lunak tersebut kemudian jumlah sumber daya yang diperlukan lalu mempertimbangkan dari kemampuan keuangan pelanggan atau dikenal dengan *Guesstimate* (Marchewka, 2003). Menggunakan metode *Guesstimate* akan menyebabkan estimasi biaya bisa lebih atau kurang dari biaya sesungguhnya, sedangkan menggunakan metode *parametric* hasil dari estimasi akan lebih akurat. Pihak CV. Profile Image Studio menjelaskan ketika mereka akan mengembangkan suatu peragkat lunak tetapi mereka belum memiliki pengalaman dalam membuat perangkat lunak tersebut sebelumnya maka mereka akan sedikit kesulitan ketika menggunakan metode *Guesstimate* .

Penelitian ini bertujuan untuk mengimplementasikan metode *Artificial Neuron Network* (ANN) *Model Based on Use Case Points* (UCP) dalam menghitung estimasi *effort* dan estimasi *effort* tersebut akan digunakan untuk menghitung durasi waktu, sumber daya manusia serta biaya pengembangan sistem. Metode *Artificial Neuron Network* (ANN) *Model Based on Use Case Points* (UCP) sendiri

UNIVERSITAS BRAWIJAYA

merupakan metode yang berdasar dari metode *Use Case Point* yang diperkenalkan oleh Gustav Karner pada tahun 1993. Pada metode *Artificial Neuron Network* (ANN) *Model Based on Use Case Points* (UCP) akan membutuhkan tiga masukan yang mencakup ukuran perangkat lunak, produktivitas dan kompleksitas proyek. Ukuran dan nantinya produktivitas perangkat lunak akan diperkirakan menggunakan *Use Case Points* (UCP).

Metode *Artificial Neuron Network* (ANN) *Model Based on Use Case Points* (UCP) dipilih untuk digunakan karena metode tersebut lebih baik dari metode *Use Case Point* (UCP) dan metode *Multiple Linier Regression* berdasarkan penelitian yang dilakukan oleh Ali Bou Nassif dan Luiz Fernando Capretz yang berjudul "*Estimating Software Effort Using an ANN Model Based on Use Case Points*". Dari penilitian disimpulkan bahwa metode ANN mengungguli metode *Multiple Linier Regression* dan Use Case Point jika dilihat pada kriteria *Mean of Magnitude of Error Relative to the estimate* (MMER). Nilai MMER ANN adalah 0,49. Nilai tersebut lebih kecil jika dibandingkan dengan nilai MMER dari Multiple linear regression sebesar 0,57 dan Use case Point sebesar 0,99. Hal tersebut yang menyebabkan dipilihnya metode *Artificial Neuron Network* (ANN) *Model Based on Use Case Points* (UCP) ini. Berdasarkan penjelasan tersebut, penelitian ini akan mengimplementasikan metode *Artificial Neuron Network* (ANN) *Model Based on Use Case Points* (UCP) di CV. Profile Image Studio, yang nantinya hasil dari penelitian ini akan dijadikan masukan bagi pihak CV. Profile Image Studio dalam menghitung biaya.

1.2 Rumusan masalah

Setelah sebelumnya dijelaskan mengenai latar belakang dari penelitian ini, maka masalah yang akan diteliti adalah:

1. Bagaimana hasil estimasi *effort*/usaha dengan menggunakan metode *artificial neural network* (ANN) model *based on use case point* ?
2. Bagaimana hasil estimasi jumlah sumber daya manusia dengan menggunakan metode *artificial neural network* (ANN) model *based on use case point* ?
3. Bagaimana hasil estimasi waktu dengan menggunakan metode *artificial neural network* (ANN) model *based on use case point* ?
4. Bagaimana hasil implementasi metode *artificial neural network* (ANN) model *based on use case point* dalam menghitung biaya pembuatan perangkat lunak ?

1.3 Tujuan

Tujuan dari dilakukannya penelitian ini adalah:

1. Memperoleh hasil estimasi *effort*/usaha dengan menggunakan metode *artificial neural network* (ANN) *based on use case point*.
2. Memperoleh hasil estimasi jumlah sumber daya manusia (*staff*) dengan menggunakan metode *artificial neural network* (ANN) *based on use case point*.

3. Memperoleh hasil estimasi waktu dengan menggunakan metode *artificial neural network* (ANN) *based on use case point*.
4. Memperoleh hasil implementasi metode *artificial neural network* (ANN) *based on use case point* dalam menghitung biaya pembuatan perangkat lunak.

1.4 Manfaat

1. Menambah bahasan keilmuan mengenai estimasi biaya menggunakan metode *Artificial Neuron Network* (ANN) *Model Based on Use Case Points* (UCP)
2. Risiko terjadinya estimasi biaya berlebih dan estimasi yang kurang bisa dihindari.
3. Memberikan masukan berupa informasi yang bisa digunakan sebagai bahan pertimbangan dalam masalah mengalokasikan biaya pengembangan perangkat lunak melalui metode estimasi *artificial neural network based on use case point*.

1.5 Batasan masalah

Supaya pembahasan lebih terarah dan tidak menyimpang batasan masalah dalam penelitian ini adalah:

1. Pembuatan *use case diagram dan use case scenario* dari sistem DBA *ticketing* dan sistem pintu air berdasarkan hasil wawancara kepada tim pengembang, manajer proyek dan pengamatan oleh penulis.
2. Nilai pada *environmental factor* berasal dari hasil penilaian oleh manajer proyek.
3. Alokasi distribusi *effort* mengacu pada *guideline* oleh penelitian Kassem Saleh (2011).

1.6 Sistematika pembahasan

Berikut merupakan deskripsi struktur skripsi pada masing-masing bab:

Bab I: Pendahuluan

Menjelaskan tentang latar belakang penilitian, rumusan masalah, batasan masalah penilitian, tujuan penelitian, manfaat penelitian, dan sistematika penulisan.

Bab II : Landasan Kepustakaan

Berisi penjelasan referensi penelitian, baik berupa paper maupun buku yang dijadikan acuan untuk melaksanakan penelitian ini

Bab III: Metodologi

Menjelaskan mengenai metode, sumber data, metode analisis yang digunakan dalam penelitian ini serta hasil dari penelitian ini.

Bab IV: Pengumpulan Data

Menjelaskan tentang data yang berhasil dikumpulkan oleh yaitu *use case* diagram, *use case* skenario, hasil penilaian yang diisi manajer proyek, serta biaya, SDM, waktu pengembangan perangkat lunak dari perushaan.

Bab V : Pembahasan

Membahas tentang pengimplementasia metode *Artificial Neuron Network* (ANN) *Model Based on Use Case Points* (UCP) yang hasilnya digunakan untuk mendapatkan nilai *effort*, estimasi durasi waktu pengembangan sistem, estimsai sumber daya manusia (staff) serta estimasi biaya pengembangan sistem dan total biaya pengembangan sistem.

Bab VI : Penutup

Bab ini menjelaskan bagaimana kesimpulan penelitian ini berdasarkan dari pokok permasalahan yang ada dan saran untuk mengembangkan penelitian ini kedepan.

BAB 2 LANDASAN KEPUSTAKAAN

Landasan kepustakaan berisi pembahasan mengenai teori, konsep, model, atau metode yang berkaitan dengan masalah atau topik penelitian. Kajian pustaka menjelaskan secara umum penelitian-penelitian terdahulu yang berhubungan dengan topik skripsi dan menunjukkan persamaan dan perbedaan skripsi tersebut terhadap penelitian terdahulu yang dituliskan. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian. Landasan teori merupakan penjelasan teori yang akan digunakan sebagai dasar penelitian dan untuk menunjang penulisan skripsi yang berjudul "Implementasi Metode *Artificial Neuron Network Based On Use Case Point* dalam menghitung biaya (Studi Kasus CV. Profile Image Studio)".

2.1 Profil CV. Profile Image

CV. Profile Image studio adalah sebuah perusahaan yang di bangun oleh semua sebuah team yang terdiri dari tenaga profesional muda yang memiliki pengalaman bertahun - tahun di bidang industri teknologi informasi, desain dan videografi. Didirikan pada tanggal 19 Juli 2011 Profile Image Studio adalah perusahaan yang tumbuh sebagai sebuah perusahaan unik dengan kompetensi tinggi dalam kombinasi teknologi informasi , seni , desain , videografi dan pengetahuan mendalam dalam pengembangan bisnis internet.

CV. Profile image studio sendiri memiliki visi yaitu "Menjadi perusahaan terdepan yang dapat memberi nilai keunggulan bagi ekosistem bisnis di Indonesia dengan menggunakan pendekatan berbasis teknologi". Demi tercapainya visi tersebut CV. Profile image studi juga memiliki beberapa misi yaitu :

- Menjalin hubungan kerjasama dengan beraneka ragam bidang industry
- Menghadirkan potensi baru dengan merangkaipadu teknologi termutakhir
- Membangun standar industri yang ideal dalam hal pelayanan jasa teknologi informasi

Dalam upayanya membangun dunia digital di Indonesia, Profile Image Studio berupaya selalu membuat terobosan terobosan baru dalam hal memberikan solusi yang tepat bagi perusahaan yang ingin membangun bisnis secara digital. Dengan didukung oleh sumber daya manusia yang mempunyai kompetensi yang tinggi serta jaringan bisnis dalam dunia digital yang luas maka akan memberikan nilai manfaat yang sangat besar dalam memberikan kontribusi dalam keberhasilan sebuah strategi baru yang akan diimplementasikan. Dalam setiap melaksanakan projectnya Profile Image Studio memegang beberapa prinsip yitu :

1. Customer Added Value
2. Optimalisasi

3. Integritas dan Kompetensi

2.2 Manajemen

Manajemen meliputi kegiatan merencanakan, mengorganisasi, mencari sumber daya, memberi instuksi, memantau kemajuan, mengontrol, memiliki inovasi dan merepresentasi (Hughes dan Cotterell, 2002).

Manajemen adalah ilmu atau seni yang mengatur proses pemanfaatan sumber daya manusia dan sumber-sumber lainnya secara efektif dan efisien untuk mencapai tujuan tertentu (Hasibuan, 2007).

Dari pendapat di atas, maka dapat disimpulkan bahwa manajemen itu adalah proses perencanaan, pengaturan dan pengkoordinasian kegiatan - kegiatan kerja dan penggunaan sumber daya secara efeketif dan efisien agar tujuang yang diinginkan dapat tercapai.

2.3 Proyek

Proyek dapat diasumsikan sebagai sesuatu yang besar untuk ditentukan bagaimana cara untuk menyelesaikan suatu pekerjaan (Hughes dan Cotterell, 2002). Beberapa karakteristik proyek dapat disimpulkan antara lain :

1. Pekerjaan yang tidak rutin dilibatkan
2. Diperlukan perencanaan
3. Objektif yang spesifik dapat dillihat atau produk yang spesifik dapat dibuat
4. Pekerjaan diselesaikan dalam beberapa fase
5. Sumber daya yang dapat digunakan dalam proyek dibatasi
6. Proyek itu besar dan kompleks

Proyek adalah suatu usaha yang bersifat sementara untuk menghasilkan suatu produk atau layanan yang unik (Schwalbe, 2004). Proyek normalnya melibatkan beberapa orang yang saling berhubungan aktivitasnya.

Atribut suatu proyek antara lain :

1. Sebuah proyek memiliki tujuan yang khusus. Proyek harus menghasilkan suatu produk khusus, layanan dan hasil akhir.
2. Proyek bersifat sementara. Proyek memiliki awal dan akhir yang jelas
3. Proyek membutuhkan sumberdaya yang didapat dari berbagai area. Sumberdaya dapat berupa hardware, software dan sumberdaya lainnya yang dilakukan oleh pengguna sistem tersebut
4. Proyek harus memiliki pelanggan utama (primary customer) atau sponsor.
5. Proyek melibatkan ketidakpastian, karena setiap proyek bersifat unik sehingga sangat sulit untuk menentukan objektifitas proyek, mengestimasi waktu dan biaya proyek.

Setiap proyek memiliki batasan yang berbeda terhadap ruang lingkup, waktu dan biaya yang biasanya disebut sebagai triple constraint (tiga kendala) (Schwalbe, 2004). Seperti itu pula seorang project manager harus memperhatikan hal-hal penting dalam manajemen proyek :

1. Scope : apa yang ingin dicapai dalam proyek, produk atau layanan apa yang pelanggan harapkan dari proyek tersebut
2. Time : berapa lama waktu yang dibutuhkan untuk menyelesaikan proyek dan bagaimana jadwal kegiatan proyek akan dilaksanakan
3. Cost : total biaya yang dibutuhkan untuk menyelesaikan proyek

Dari beberapa beberapa pendapat di atas, maka disimpulkan bahwa proyek adalah suatu usaha yang bersifat sementara namun dibutuhkan atribut yang dapat menunjang jalannya proyek agar dapat menghasilkan produk atau hasil yang baik dan jelas. Atribut tersebut antara lain dalam hal-hal waktu, biaya dan sumber daya.

2.4 Manajemen Proyek Teknologi Infromasi

Manajemen proyek adalah suatu cara untuk menyelesaikan masalah yang harus dipaparkan oleh *user,* kebutuhan *user* harus terlihat jelas dan harus terjadi komunikasi yang baik agar kebutuhan *user* bisa diketahui (Hughes dan Cotterell, 2002). Manajemen Proyek adalah aplikasi pengetahuan, keahlian, peralatan dan teknik untuk kegiatan proyek yang sesuai dengan kebutuhan proyek (Schwalbe, 2004).

Dari pendapat diatas, maka dapat disimpulkan bahwa manajemen proyek adalah kegiatan merencanakan, mengatur, mengkoordinasikan seluruh sumber daya, biaya, waktu untuk menghasilkan suatu hasil yang bersifat sementara atau suatu produk yang unik dan sesuai dengan kebutuhan yang diminta.

Disimpulkan juga bahwa, manajemen proyek teknologi informasi adalah merencanakan, menyusun, mengorganisasikan seluruh sumber daya, waktu, dan biaya untuk membuat suatu rekayasa manusia terhadap proses penyampaian informasi kepada penerima akan lebih cepat, lebih luas sebarannya dan lebih lama penyimpanannya

2.5 *Project Life Cycle*

Menurut Marchewka (2003), *Project Life Cycle* (PLC) merupakan kumpulan tahapan logis dari awal hingga akhir proyek untuk mendefinisikan, membangun, dan memberikan produk dari proyek. Estimasi biaya dilakukan pada tahap *Plan Project* yang termasuk kedalam salah satu tahap *Project Life Cycle*. Tahapan yang terdapat dalam siklus hidup proyek ini adalah *Define Project Goal*, *Plan Project*, *Execute Project Plan*, *Close Project*, dan *Evaluate Project*, yang diuraikan sebagai berikut Marchewka (2003):

1. *Define Project Goal* merupakan tahapan yang mendefinisikan seluruh tujuan proyek. Tujuan proyek terfokus pada penyediaan nilai bisnis kepada

perusahaan. Dengan baiknya tujuan yang ditetapkan memberikan tim proyek sebuah tujuan yang jelas dan menggerakkan tim ke fase lain proyek.

2. *Plan Project* merupakan tahapan perencanaan proyek yang berisi tentang aktivitas yang dilakukan di dalam proyek yang dikerjakan. Alasan apa yang membuat aktivitas tersebut dilakukan di dalam proyek yang dikerjakan kemudian menjelaskan cara untuk melakukan aktivitas tersebut lalu siapa yang akan mengerjakan aktivitas tersebut di dalam proyek yang dikerjakan kemudian berapa lama waktu yang akan digunakan untuk melakukan aktivitas tersebut serta menjelaskan cara untuk mengestimasi biaya serta menjelaskan kriteria kesuksesan dari proyek yang dikerjakan.
3. *Execute Project Plan* merupakan tahapan untuk melaksanakan proyek sesuai dengan tujuan dan perencanaan yang telah ditentukan. Pada akhir tahap ini, tim proyek menyampaikan atau mengimplementasikan produk kepada organisasi. Pada tahap ini, terdapat siklus hidup pengembangan sistem (*Systems Development Life Cycle*).
4. *Close Project* merupakan tahapan penutupan proyek untuk memastikan bahwa semua pekerjaan telah terselesaikan sesuai dengan yang direncanakan dan disetujui oleh tim proyek dan sponsor. Tahap ini biasanya diakhiri dengan laporan akhir proyek dan mempresentasikan dokumen yang berisi bahwa segala deliverable yang dijanjikan telah diselesaikan seperti yang telah ditetapkan kepada pelanggan.
5. *Evaluate Project* merupakan tahapan untuk mengevaluasi proyek untuk menentukan apakah proyek berhasil memenuhi tujuan proyek. Evaluasi proyek dapat dilakukan melalui dokumentasi pengalaman tim proyek yang dijadikan bahan pelajaran. Berdasarkan hal tersebut, maka dapat dilakukan perbaikan pada proyek selanjutnya.

2.6 *System Development Life Cycle*

Dalam fase *Execute Project Plan* pada *Project Life Cycle*, terdapat *Systems Development Life Cycle* (*SDLC*). SDLC menjelaskan tahapan dalam proses pengembangkan sistem informasi (Schwalbe, 2012). SDLC adalah bagian dari *Project Life Cycle* (PLC) sebab ketika fase eksekusi beberapa aktivitas untuk proses pengembangan sistem informasi (Marchewka, 2003). Terdapat 5 fase dasar yang pada umumnya terdapat dalam pengembangan sistem yaitu *Planning*, *Analysis*, *Design*, *Implementation*, dan *Maintenance and Support* (Marchewka, 2003).

1. Fase *Planning* mencakup identifikasi dan penanganan masalah atau kesempatan dan menggabungkan manajemen proyek dengan proses dan aktivitas pengembangan sistem. Pada tahap ini dilakukan proses perencanaan resmi yang menjamin tujuan, cakupan, anggaran, penjadwalan, teknologi, proses pengembangan, metode, dan *tool* sudah tersedia.
2. Fase *Analysis* bertujuan untuk menyelidiki masalah atau peluang. Fase ini dilakukan dengan mengidentifikasi dan mendokumentasikan masalah atau

hambatan pada sistem yang sedang digunakan oleh pelanggan. Kebutuhan spesifik sistem baru diidentifikasi dan didokumentasikan.

3. Fase *Design* bertujuan untuk merancang arsitektur untuk mendukung sistem informasi baru dengan menggunakan kebutuhan yang telah diperoleh pada fase Analysis. Arsitektur yang dimaksud mencakup perancangan jaringan, konfigurasi *hardware*, konfigurasi database, perancangan antarmuka pengguna, dan program aplikasi.
4. Fase *Implementation* mewakili konstruksi sistem, pengujian serta istalasi.
5. Fase *Maintenance and Support* mencakup pemeliharaan dan perbaikan yang dilakukan untuk menangani perubahan sistem.

2.7 Metode *Guesstimate*

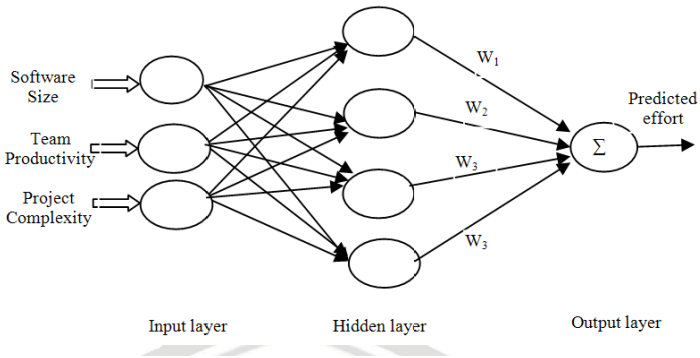
Estimasi dengan metode *guesstimate* bukanlah cara terbaik untuk menghitung anggaran biaya proyek. Sayangnya, banyak manajer proyek yang tidak berpengalaman cenderung memperkirakan suatu anggaran biaya dengan didasarkan pada intuisi dan bukan melalui bukti karena dirasa lebih cepat dan mudah. Hal tersebut harus disikapi dengan hati-hati ketika manajer proyek mengutip rancangan anggaran biaya dari catatan pembuatan proyek sebelumnya, karena seringkali mereka terlalu optimis dan dapat mengakibatkan waktu berjam-jam terbuang dan mengurangi kualitas (Marchewka, 2003).

2.8 *Artificial Neuron Network* (ANN)

Artificial Neural Network (ANN) atau dalam bahasa Indonesia disebut jaringat saraf tiruan adalah jaringan yang terdiri dari neuron buatan atau kelenjar yang mengemulsi neuron biologis (A. B. Nassif, 2012). ANN dapat dilatih untuk digunakan dalam memperkirakan, memetakan input ke output atau untuk mengklasifikasikan output. Topologi yang paling menonjol dari ANN adalah jaringan *feed-forward*. Jaringan *feed-forward* biasanya direpresentasikan sebagai *input*, *hidden*, dan *output layer*. Jika *hidden layer* tidak ada, maka jenis ANN ini disebut *perceptron*.

Perceptron dapat memetakan input ke output jika hubungan antara input dan output adalah linear. Jika hubungan antara input dan output tidak linier, satu atau lebih *hidden layer* akan terbentuk diantara lapisan *input* dan *output* untuk mengakomodasi sifat non-linear. Ada beberapa tipe jaringan *feed-forward* yang memiliki *hidden layer* didalamnya, seperti *Multilayer Perceptron* (MLP), *Radial Basis Function Neural Network* (RBFNN) dan *General Regression Neural Network* (GRNN). Di dalam MLP sendiri mengandung setidaknya satu *hidden layer* dan setiap input diwakili oleh sebuah neuron. Jumlah neuron yang tersembunyi bervariasi dan dapat ditentukan dengan trial and error sehingga meminimalkan kesalahan. Dalam penelitian yang dilakukan oleh A. B. Nassif (2012), jenis MLP digunakan untuk memprediksi *effort* atau usaha perangkat lunak berdasarkan ukuran perangkat lunak yang dihitung berdasarkan metode *use case point* (UCP), produktivitas tim, dan kompleksitas proyek. Gambar 2.18 menunjukkan

bagaimana arsitektur ANN yang digunakan dalam penelitian oleh A. B. Nassif (2012) dengan tiga input dan empat neuron tersembunyi.



Gambar 2.1 Arsitektur dari ANN

2.9 Metode *Artificial Neuron Network Model Based On Use Case point*

Berdasarkan penelitian yang dilakukan oleh Ali Bou Nassif dan Luiz Fernando Capretz yang berjudul "*Estimating Software Effort Using an ANN Model Based on Use Case Points*". Dari penilitian disimpulkan bahwa metode ANN mengungguli metode *Multiple Linier Regression* dan Use Case Point jika dilihat pada kriteria *Mean of Magnitude of Error Relative to the estimate* (MMER). Nilai MMER ANN adalah 0,49. Nilai tersebut lebih kecil jika dibandingkan dengan nilai MMER dari Multiple linear regression sebesar 0,57 dan Use case Point sebesar 0,99. Hal tersebut yang menyebabkan dipilihnya metode *Artificial Neuron Network* (ANN) *Model Based on Use Case Points* (UCP) ini.

Pada metode ini untuk menghitung *effort* didapatkan dari ukuran perangkat lunak, produktifitas dan kompleksitas. Ukuran perangkat lunak diperkirakan berdasarkan metode UCP, nilai produktifitas diperkirakan dengan menetapkan nilai skala antara 0 sampai 5 pada setiap *Environmental factor* yang terdapat pada Tabel 2.4 lalu dikalikan dengan bobot dari setiap faktor, jadi secara untuk menghitung nilai produktifitas dapat mengunakan persamaan berikut :

$$Productivity = \sum_{i=1}^{8} \mathrm{n} \times Wi \qquad (2.2)$$

Sedangkan untuk nilai Kompleksitas dapat diperkirakan berdasarkan lima *level* (yaitu sebagai berikut :

Level 1: Tim proyek terbiasa dengan jenis proyek ini dan tim telah mengembangkan proyek serupa sebelumnya. Jumlah dan tipe *interface* yang sederhana. Proyek akan dipasang dalam kondisi normal dimana keamanan tinggi atau faktor keamanan tidak diperlukan, Sekitar 20% dari desain atau bagian proyek berasal dari proyek yang serupa. Bobot kompleksitas *Level* 1 adalah 1.

Level 2: Sekitar 10% dari desain atau bagian proyek berasal dari proyek yang serupa. Bobot kompleksitas *Level* 2 adalah 2.

Level 3: Proyek tidak dikatakan sederhana, tidak rumit. Pada *level* ini, teknologi, *interface*, kondisi instalasi yang normal. Tidak ada bagian dari proyek yang mengimplementasi proyek sebelumnya. Bobot kompleksitas *Level* 3 adalah 3.

Level 4: Proyek dipasang pada topologi / arsitektur yang rumit. Memiliki jumlah variable yang besar dan *interface* yang rumit. Berat *Level* 4 kompleksitas adalah 4.

Level 5: Proyek dipasang pada topologi / arsitektur yang rumit. Memiliki jumlah variable yang besar dan *interface* yang rumit. Memiliki keamanan khusus atau keamanan tingkat tinggi. Bobot kompleksitas *Level* 5 adalah 5.

Untuk menghitung *effort* pada metode *Artificial Neuron Network Based On Use Case point* menggunakan persamaan :

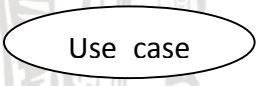
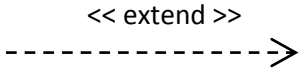
Effort = 3.53 + (0.88 x Size) – (0.009 x *Productivity*) + (0.31 x Complexity) (2.3)

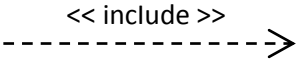
Hasil *effort* yang didapatkan memiliki satuan *person-hours.*

2.10 Pemodelan Use Case

Pemodelan *Use case diagram* berfungsi untuk menggambarkan bagaimana interaksi antara pengguna dengan sistem. Terdapat aktor, *use case* dan sistem di dalam *use case diagram*. *Use case* merepresentasikan kebutuhan fungsional dari sistem. Aktor adalah peran yang di perankan oleh pengguna dari suatu sistem. Deskripsi fungsionalitas yang berada di dalam sistem sebagai aktor yang berinteraksi satu dengan yang lainnya yang ditulis menggunakan kata kerja terdapat di dalam *use case* (Object Management Group, 2005).

Tabel 2.1 Simbol pada *use case diagram*

No	Nama	Simbol	Keterangan
1	*Use Case*		Fungsionalitas pada sistem.
2	Aktor		Berupa orang yang berinteraksi dengan sistem.
3	Asosiasi		Hubungan interaksi antara *use case* dengan aktor.
4	*Extend*		Relasi *use case* yang bisa menambah tingkah lakunya kepada *use case* lainnya.

5	*Generalization*		Generalisasi dan spesialisasi antar *use case*.
6	*Include*		Akan menyebabkan *use case* menjadi bagian dari proses suatu *use case* yang lainnya.

Pada tabel 2.1 menjelaskan deskripsi mengenai simbo-simbol yang ada pada *use case* dan keterangan dari masing-masing symbol tersebut. *Use case scenario* merupakan penjelasan secara tekstual dari sekumpulan skenario interaksi. *Use case scenario* berisi tentang bagaimana *use case* digunakan, dan juga tindakani spesifik yang dilakukan oleh pengguna. *Use case scenario* terdiri dari (primary *actor*) yang merupakan aktor yang terlibat dalam skenario, (*precondition)* pra kondisi sistem yang merupakan kondisi yang harus terpenuhi sebelum sebuah *use case* bisa dieksekusi, (*main of basic flow*) skenario keberhasilan utama yang merupakan akvitas yang mengarah pada skenario yang berhasil agar tujuan dari aktor dapat terpenuhi, kemudian (*alternative flow)* jalur alternative yang merupakan jalur alternatif jika aktor dengan sistem terdapat pilihan yang lain maupun skenario gagal, dan (*postcondition)* kondisi akhir adalah kondisi yang ditulis dengan detail dan kondisi tersebut harus terjadi ketika *use case* telah berhasil dilakukan oleh aktor (kurniawan, 2018).Tingkat kompleksitasan suatu *use case* ditetapkann dengan melihat melalui total jumlah transaksi pada *use case* skenario. Transaksi yang dihitung yaitu mulai transaksi pada *main of basic flow* dan pada *alternative flow* (Nassif *et al.,* 2010). Sebuah perjalanan dari aktor ke sistem kembali lagi ke aktor aktor atau *round trip* juga merupakan definisi dari transaksi (Jacobson *et al*, 1992).

2.11 *Use Case Point*

Metode *Use Case Point* menggunakan dasar *use case* untuk menghitung estimasi *effort* pengembangan perangkat lunak (Anda, 2002). Pada tahun 1993 metode use case point (UCP) pertama kali dijelaskan oleh Gustav Karner. Model ini digunakan untuk estimasi biaya perangkat lunak berdasarkan diagram use case. *Unadjusted Use Case Weight* (UUCW) menunjukkan bagaimana tingkat kompleksitas suatu *use case* yang dilihat dari jumlah transaksinya. Transaksi dapat dihitung dengan cara menghitung jumlah langkah yang ada pada *basic flow* dan *alternative flow* yang ada didalam sebuah use case (Kristen Ribu, 2001). Terdapat tiga kategori berdasarkan jumlah dari transaksi yang terjadi dalam sebuah *use case.* Kategori tersebut yaitu sederhana, medium atau rata-rata, dan kompleks. *Unadjusted Use Case Weight* (UUCW) dihitung melalui penjumlahan bobot setiap *use case* menggunakan persamaan sperti berikut:

$$UUCW = \sum_{i=1}^{3} \mathrm{n} * Wi \qquad (2.4)$$

n merupakan jumlah *use case*, *i* merupakan urutan tertentu, sedangkan W*i* merupakan bobot *use case* urutan ke-*i* (lihat Tabel 2.3)

Tabel 2.2 Kategori kompleksitas *use case*

Jumlah transaksi dalam *use case*	Komplesitas	Bobot
Kurang dari 4 transaksi	Sederhana	5
4 hingga 7 transaksi	Medium	7
Lebih dari 7 transaksi	Kompleks	15

Aktor setiap *use case* dikategorikan dalam sederhana, rata-rata, atau kompleks (Clemmon, 2006). Aktor dengan kategori sederhana yaitu aktor mewakili sistem lain dengan API yang telah ditentukan. Aktor dengan kategori rata-rata adalah aktor mewakili sistem lain yang berinteraksi melalui protokol seperti TCP/IP. Aktor dengan kategori kompleks merupakan orang yang berinteraksi dengan sistem melalui *Graphical User Interface* (GUI). Seperti yang terlihat pada table 2.3 di bawah.

Tabel 2.3 Deskripsi Kategori aktor

Kategori Aktor	Deskripsi	Nilai Bobot
Sederhana	Aktor berinteraksi menggunakan *Application Programming Interface* (API).	1
Rata-rata	Aktor berinteraksi menggunakan commend line.	2
Kompleks	Aktor berinteraksi melalui *Graphical User Interface*.	3

Untuk menghitung *Unadjusted Actor Weight* (UAW) dengan cara berapa banyak aktor dari masin-masing kategori kemudian dikalikan dengan bobot dari masing-masing kategori seperti pada persamaan 2.4 di bawah ini.

$$UAW = \sum_{i=1}^{3} \mathrm{m} * Ci \qquad (2.5)$$

m merupakan jumlah aktor, *i* merupakan urutan tertentu, sedangkan Ci merupakan bobot aktor urutan ke-i.

Setelah nilai dari *Unadjusted Use Case Weight* (UUCW) dan *Unadjusted Actor Weight* (UAW) didapatkan maka untuk mendapatkan nilai *Unadjusted Use Case Points* (UUCP) dapat dihitung dengan menjumlahkan *Unadjusted Use Case Weight* (UUCW) dengan *Unadjusted Actor Weight* (UAW) seperti pada persamaan 2.5 berikut :

UUCP = UUCW + UAW (2.5)

Nilai dari *Technical Complexity Factor* (TCF) dan *Environmental Complexity Factors* (ECF) dari proyek harus dihitung juga (Nassif Ali, 2011) karena, merupakan

salah satu faktor untuk memperkirakan ukuran perangkat lunak dengan memperhitungkan pertimbangan teknis. Untuk mendapatkan nilai dari *Technical Complexity Factor* (TCF) maka nilai faktor ditentukan mulai dari 0 hingga 5 kemudian dikalikan dengan bobot dari masing-masing faktor. Nilai 0 berarti faktor tidak berpengaruh dengan proyek, sedangkan nilai 3 berarti memiliki pengaruh yang biasa saja, dan nilai 5 berarti faktor tersebut berpengaruh kuat (Kristen Ribu, 2001). Apabila ragu-ragu dalam memberi nilai dan ketika jumlah *use case* yang ada kurang dari 50 maka dapat diberikan nilai 3 pada *technical factor* (Ani dan Basri, 2013). Dengan jumlah *use case* pada sistem kurang dari 50 *use case*, berarti sistem yang dikembangkan tidak terlalu rumit, dan tingkat kompleksitasnya dianggap rata-rata (Ani & Basri, 2013). Pada persamaan 2.6 berikut merupakan persamaan dari *Technical Complexity Factor* (TCF) :

$$TCF = 0.6 + 0.01 * (\sum_{i=1}^{13} Ti \times Wi) \quad (2.6)$$

Tabel 2.4 *Technical Factors*

Ti	Faktor	Bobot
T_1	Sistem Terdistribusi	2
T_2	Performa (Respon time)	1
T_3	Efisiensi dari *user interface*	1
T_4	Komplesitas proses	1
T_5	Penggunaan kembali kode program	1
T_6	Kemudahan instalasi	0.5
T_7	Kemudahan pengoperasian	0.5
T_8	Dapat diaplikasikan diberbagai platform	2
T_9	Kemudahan perawatan	1
T_{10}	Konkurensi	1
T_{11}	Fitur Keamanan	1
T_{12}	Ketergantungan pada pihak Ketiga	1
T_{13}	Ketersedian bantuan	1

Tabel 2.4 merupakan faktor yang terdapat *pada technical factor* serta bobot dari masing-masing faktornya. Nilai *Environmental Complexity Factor* (ECF) dhitung dengan menetapkan nilai antara 0 sampai 5 di setiap faktor (lihat Tabel 2.5) kemudian dikalikan dengan bobot dari masing-masing faktor. Nilai 0 berarti faktor tersebut tidak relevan dengan sistem yang dibangun, dan nilai 5

berarti faktor tersebut sangat relevan dengan sistem yang dibangun. Adapun nilai pada setiap faktor ditetapkan oleh manajer proyek (Ribu, 2001). Persamaan untuk menghitung nilai *Environmental Complexity Factor* (ECF) yaitu seperti berikut :

$$EF = (\sum_{i=1}^{8} Ei \times Wi) \quad (2.7)$$

$$ECF = 1.4 + (-0.303 * (\sum_{i=1}^{8} Ei \times Wi) \quad (2.8)$$

Tabel 2.5 *Environmental Factor*

E_i	Faktor	Bobot
E_1	Mengenal metode yang digunakan dalam membuat proyek	1.5
E_2	Pengalaman membuat Aplikasi	0.5
E_3	Pengalaman menggunakan pemrograman berorientasi obyek	1
E_4	Menguasai Kemampuan Analisis	0.5
E_5	Motivasi	1
E_6	Stabilitas kebutuhan proyek	2
E_7	Pekerja paruh waktu	-1
E_8	Kesulitan dalam Bahasa Pemrogaman	-1

Pada tabel 2.5 merupakan faktor yang terdapat pada *environmental factor* serta bobot dari masing-masing faktornya dan berikut deskripsi tiap faktor pada *environmental factor* beserta nilainya.

Mengenal metode yang digunakan dalam membuat proyek. Faktor ini untuk mengukur pengalaman tim dengan metode yang digunakan pada proyek

a. Nilai 0: Tim tidak terbiasa dengan proses pengembangan aplikasi.
b. Nilai 1: Tim memiliki pengetahuan teoritis terhadap proses pengembangan.
c. Nilai 2-3: Satu atau lebih anggota tim telah menggunakan metode tersebut sekali atau beberapa kali.
d. Nilai 3-4: Setidaknya setengah dari anggota tim memiliki pengalaman menggunakan metode tersebut pada proyek yang berbeda.
e. Nilai 5: Seluruh anggota tim memiliki pengalaman menggunakan metode tersebut pada beberapa proyek berbeda.

Pengalaman Aplikasi. Faktor ini mengindikasikan pengalaman dalam pembuatan sistem dengan jenis ini atau sistem dengan jenis yang berbeda.

a. Nilai 0 : Semua anggota tim tidak memiliki pengalaman
b. Nilai 1-2: Beberapa anggota tim memiliki pengalaman, contohnya 1 sampai 1,5 tahun, dan anggota tim lainnya pemula

c. Nilai 3 : Semua anggota tim memiliki pengalaman lebih dari 1,5 tahun
d. Nilai 4 : Hampir semua anggota tim memiliki pengalaman 2 tahun
e. Nilai 5 : Semua anggota tim berpengalaman lebih dari 2 tahun

Pengalaman menggunakan pemrograman berorientasi objek. Faktor ini mengukur pengalaman anggota tim dalam menggunakan pemrograman berorientasi obyek.

a. Nilai 1: Semua anggota tim memiliki kurang dari 1 tahun pengalaman
b. Nilai 2-3: Semua anggota tim memiliki 1 sampai 1,5 tahun pengalaman.
c. Nilai 4: Hampir semua anggota tim memiliki lebih dari 2 tahun pengalaman
d. Nilai 5: Semua pengembang berpengalaman lebih dari 2 tahun.

Menguasai Kemampuan Analis. Faktor yang mengukur pengalaman dalam menganlisis kebutuhan dan pemodelan.

a. Nilai 0: Analis utama adalah seorang pemula.
b. Nilai 1-2: Pengalaman dari sedikit proyek.
c. Nilai 3-4 : Setidaknya 2 tahun pengalaman dari beberapa proyek.
d. Nilai 5: Setidaknya 3 tahun pengalaman dengan beragam proyek

Motivasi. Faktor ini untuk bagaimana motivasi didalam tim pengembang.

a. Nilai 0: Tidak termotivasi.
b. Nilai 1-2: Sedikit motivasi.
c. Nilai 3-4: Tim termotivasi melakukan pekerjaan baik.
d. Nilai 5: Tim sangat termotivasi dan terinspirasi

Stabilitas kebutuhan proyek. Faktor ini mengukur tingkatan perubahan kebutuhan dan ketidakpastian kebutuhan sistem.

a. Nilai 0: Kebutuhan sangat tidak stabil, perubahan secara terus-menerus.
b. Nilai 1-2: Kebutuhan tidak stabil. Pelanggan meminta berbagai perubahan dilakukan pada berbagai selang waktu.
c. Nilai 3-4: Kestabilan kebutuhan secara menyeluruh, namun masih membutuhkan perubahan kecil.
d. Nilai 5: Kebutuhan yang selalu stabil selama pengembangan aplikasi.

Kesulitan dalam Bahasa Pemrogaman. Faktor ini mengukur tingkat kesusahan yang dialami tim saat menggunakan bahasa pemrograman yang dipilih.

a. Nilai 0: Semua anggota tim adalah *programmer* berpengalaman.
b. Nilai 1: Hampir semua anggota tim memiliki pengalaman lebih dari 2 tahun.
c. Nilai 2: Semua anggota tim memiliki lebih dari 1,5 tahun pengalaman.
d. Nilai 3: Hampir semua anggota tim memiliki lebih dari 1 tahun pengalaman.
e. Nilai 4: Sedikit anggota tim yang berpengalaman, contohnya 1 tahun, dan anggota tim lainnya masih pemula.
f. Nilai 5: Semua anggota tim merupakan pemula.

Nilai *Adjusted Use Case Points* (UCP) dapat ditentukan dengan cara mengalikan nilai *Unadjusted Use Case Points* (UUCP) dengan nilai *Technical Complexity Factors* (TCF) dan dengan nilai *Environmental Complexity Factors* (ECF) sebagaimana persamaan 2.9:

$$UCP = UUCP \times TCF \times ECF \quad (2.9)$$

2.12 *Guideline* Distribusi *Effort*

Effort disitribusikan ke setiap aktivitas proses pengembangan perangkat lunak sesuai dengan panduan distribusi *effort* (Saleh K, 2011). Persentase distribusi usaha pada hasil penelitian Saleh dapat dilihat pada tabel 2.1 berikut.

Tabel 2.6 Distribusi *Effort*

Aktivitas	***%Effort***
Software Phases	
Requirements	7.5
Spesifications	7.5
Design	10
Implementation	10
Integration Testing	7.5
Acceptance & Deployment	7.5
Ongoing life-cycle activities	
Project Management	8.34
Configuration Management	4.16
Quality Assurance	8.34
Documentation	4.16
Training and support	4.16
Evaluation and Testing	20.84

Dari nilai effort setiap aktivitas pada tabel 2.1 nantinya dapat digunakan untuk menghitung jumlah staff yang bekerja dalam setiap aktivitas serta biaya yang dikeluarkan dalam setiap aktivitas.

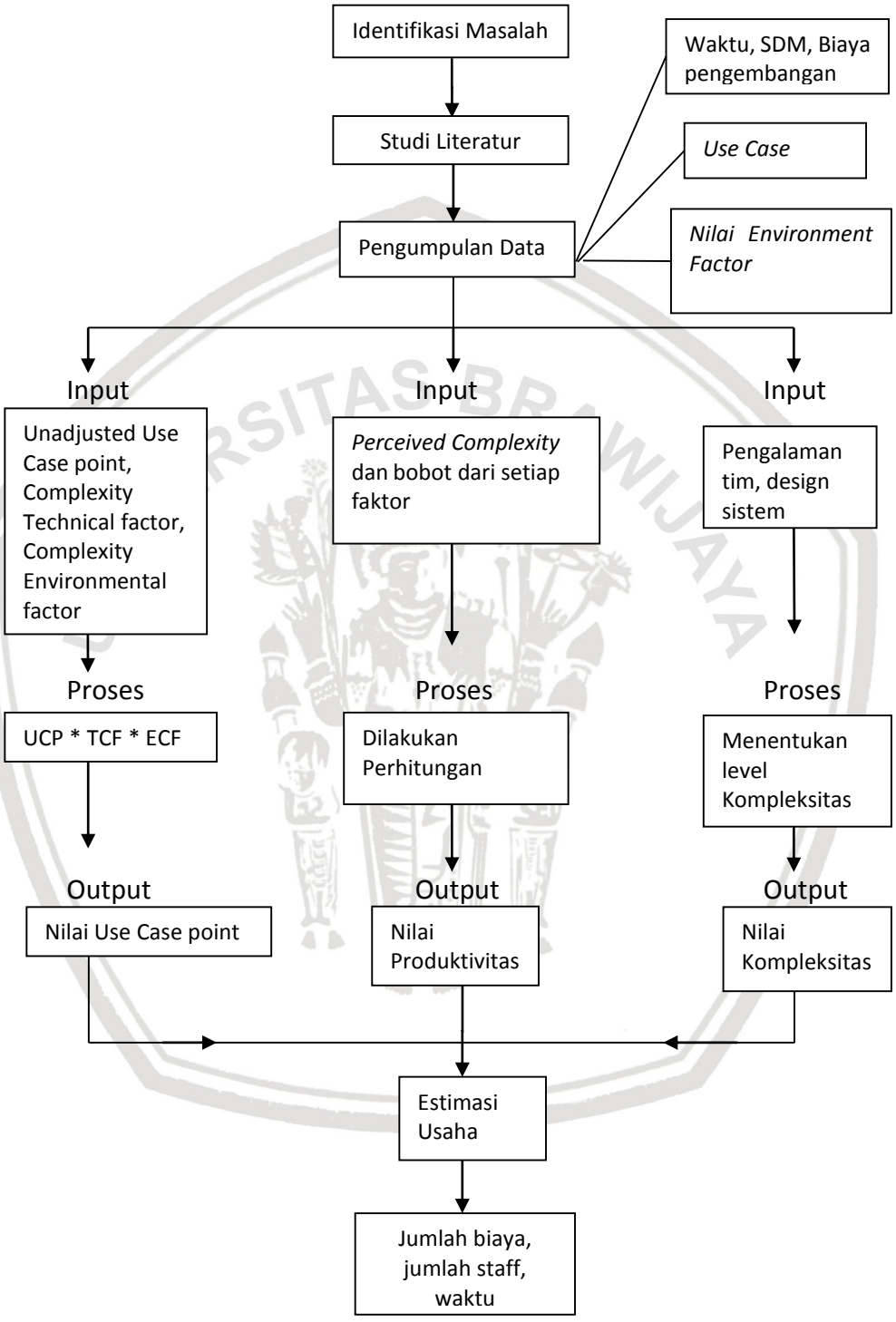
2.13 Waktu Pengembangan Perangkat Lunak

Pada penelitian Kassem Saleh (2011) juga menjelaskan untuk menghitung durasi pengembangan perangkat lunak dapat dihitung dengan cara mengkalikan effort rate dengan nilai use case point kemudian dibagi dengan jumlah jam kerja seperti persamaan dibawah ini.

$$((p \times UCP)/h) \text{ days} \quad (2.1)$$

p merupakan Effort Rate yaitu bernilai 8,2 person hours (Pribadi Apol, 2014), UCP merupakan nilai Use Case Point dan h merupakan jumlah jam kerja dalam sehari, karena jam kerja di Indonesia rata-rata adalah 8 jam maka nilai h adalah 8.

BAB 3 METODOLOGI



Metode penelitian yang dilakukan dimulai dengan mengidentifikasi masalah yang ada pada tempat penelitian, kemudian dilakukan studi pustaka untuk menentukan metode yang tepat. Tahapan berikutnya setelah masalah teridentifikasi dan menemukan metode yang tepat yaitu mengumpulkan data yang dibutuhkan untuk menentukan nilai estimasi *effort,* nilai estimasi biaya, estimasi jumlah sumber daya manusia dan estimasi biaya total pengembangan. Setelah itu dilakukan perbandingan antara hasil implementasi metode *ANN model based on use case point* dengan hasil milik perusahaan.

3.1 Identifikasi Masalah dan Menentukan Metode Estimasi

Pada tahap identifikasi masalah, wawancara dilakukan kepada perusahaan. Berdasarkan hasil wawancara dengan direktur operasional CV. Profile Image Studio, CV. Profile Image Studio sendiri dalam melakukan estimasi biaya pengembangan perangkat lunak dengan cara melihat dari tingkat kompleksitas suatu perangkat lunak tersebut kemudian jumlah sumber daya yang diperlukan lalu mempertimbangkan dari kemampuan keuangan pelanggan atau biasa disebut *guesstimate*, jadi CV. Profile Image Studio belum menggunakan metode *parametric* untuk menghitung estimasi biaya perangkat lunak yang mereka buat. Berdasarkan hasil wawancara dengan pihak CV. Profile Image Studio terjadi ketidaksesuaian dalam mengestimasi biaya dan waktu.

3.2 Studi Pustaka

Tahapan ini mengumpulkan bebagai referensi baik dari *paper*, jurnal, ataupun buku guna mendapatkan teori dasar sebagai dasar dillakukannya penelitian ini. Beberapa teori pendukung penelitian didapatkan pada tahapan ini adalah sebagai berikut:

1. Profil CV. Profile Image Studio sebagai tempat penelitian. Penelitian sebelumnya terkait metode estimasi biaya.
2. Teori pendukung mengenai manajemen proyek sistem informasi, daur hidup proyek, daur hidup perangkat lunak, estimasi biaya perangkat lunak, *work breakdown structure*, dan *gantt chart*.
3. Metode *Artificial Neural Netowrk based on Use Case Point* mencakup deskripsi metode dan cara perhitungannya.

3.3 Pengumpulan Data

Pada tahap pengumpulan data penulis melakukan pengumpulan data yang bersumber dari manajer proyek di CV. Profile Image Studio. Penulis melakukan pengumpulan data dengan cara observasi, wawancara dan membagikan lembar penilaian. Dari hasil pengempulan data didapatkan data berupa waktu, sumber daya manusia yang digunakan, serta biaya yang keluarkan untuk pengembangan sistem dan juga nilai pada *environmental factor*. Data yang berhasil dikumpulkan

kemudian akan digunakan untuk menghitung estimasi effort dengan metode *ANN model based on use case point*.

3.3.1 Observasi

Penulis melakukan pengamatan langsung di lapangan guna mengumpulkan data penelitian. Observasi dilaksanakan dengan melihat sistem. Hasil observasi akan digunakan untuk membuat diagram *use case* dan *use case* skenario. *Use case* skenario yang didapatkan akan digunakan untuk menghitung estimasi effort dengan metode *ANN model based on use case point*.

3.3.2 Wawancara

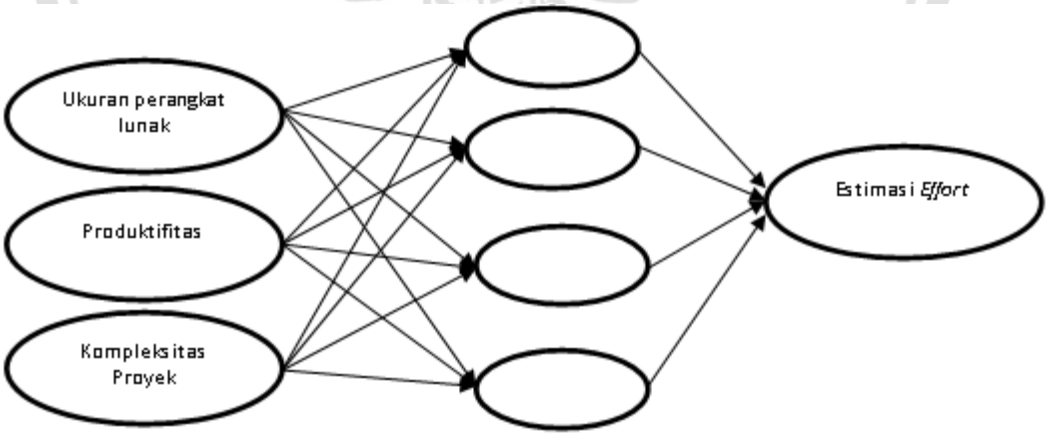
Peneliti melakukan wawancara bersama dengan narasumber. Pihak yang menjadi narasumber penelitian adalah manajer proyek serta manajer operasional CV. Profile Image studio. Hasil dari wawancara digunakan untuk membuat *use case diagram*, menentukan jumlah sumber daya manusia, durasi waktu serta total biaya pengembangan.

3.3.3 Lembar Penilaian

Pada tahap ini salah satu pengambilan data dilakukan dengan membagikan lembar penilaian environmental factor guna menghitung nilai *environmental complexity factor*. Lembar penilaian diisi oleh manajer proyek sistem di CV. Profile Image Studio.

3.4 Menghitung estimasi effort

Data yang berhasil didapatkan pada tahap pengumpulan data lalu digunakan untuk memperoleh estimasi *effort* denga mengimplementasikan metode *Artificial Nerural Network* (ANN) *based on use case point* seperti pada gambar 3.1.



Gambar 3.1 Menghitung estimasi effort

Deskripsi dari Gambar 3.1 dalam melakukan estimasi effort adalah sebagai berikut:

1. Menghitung ukuran perangkat lunak.

Ukuran perangkat lunak didapatkan dari nilai *use case point*. Nilai *use case point* didapatkan dengan mengalikan nilai *Unadjusted Use Case Point* dengan nilai *Technical Complexity Factor* dan nilai *Environmental Complexity Factor*.

2. Menghitung produktivitas.
   Menghitung produktivitas didapatkan dari nilai *enviromental complexity factor* yang mana berasal dari kompleksitas pada *enviromental factor* (*Perceived Complexity*) dikalikan dengan bobot setiap faktor, kemudian dijumlahkan seperti pada persamaan 2.7.
3. Menghitung kompleksitas.
   Kompleksitas dapat diartikan sebagai item yang memiliki dua atau lebih elemen. Kompleksitas teknis berkaitan dengan jumlah komponen produk, jumlah teknologi yang terlibat, jumlah antarmuka yang ada. Kompleksitas dapat ditentukan dengan mengacu pada lima tingkatan kompleksitas yang ada sesuai hasil penelitian A. B. Nassif (2012).

Setelah mendapatkan nilai dari ukuran perangkat lunak, nilai produktivitas dan kompleksitas dari pernagkat lunak tersebut maka untuk menghitung *effort* atau usaha dengan cara memasukkan nilai yang ada ke persamaan 2.2.

3.5 Estimasi Waktu

Estimasi waktu pengembangan sistem dihitung berdasarkan penelitian Kassem Saleh (2011). Hasil estimasi waktu didapatkan dengan dengan cara mengkalikan effort rate dengan nilai use case point kemudian dibagi dengan jumlah jam kerja seperti persamaan 2.1. Hasil estimasi waktu yang didapat pada tahapan ini kemudian digunakan untuk menghitung estimasi biaya pengembangan sistem.

3.6 Estimasi Sumber Daya Manusia

Hasil dari tahapan ini akan mendapatkan jumlah estimasi sumber daya manusia yang diperlukan dalam pengembangan sistem. Jumlah estimasi sumber daya manusia didapatkan dari pembagian nilai effort pengembangan sistem dibagi dengan estimasi waktu pengembangan sitem yang didapatkan pada tahapan sebelumnya. Jumlah estimasi waktu dan estimasi sumber daya manusia akan digunakan untuk menghitung estimasi biaya.

3.7 Estimasi Biaya

Dari estimasi waktu pengembangan sistem dan estimasi sumber daya manusia yang telah didapatkan pada tahap sebelumnya akan digunakan untuk menghitung biaya pengembangan sistem. Terdapat 2 kelompok aktivitas yaitu *Software Phases* dan *Ongoing Life Cycle Activities.* Proses penghitungan estimasi biaya pengembangan sistem diperoleh dengan mengalikan durasi pengerjaan setiap aktivitas dengan standar gaji UMK kota Malang pada tahun 2017. Biaya total

keseluruhan pengembangan perangkat lunak. Didapatkan dengan menjumlahkan biaya pada setiap aktivitas.

BAB 4 PENGUMPULAN DATA

Bab ini menjelaskan data yang telah dikumpulkan adapun data tersebut yaitu berupa waktu pengembangan sistem, sumber daya manusia yang digunakan, dan biaya pengembangan milik perusahaan. Data yang lan berupa *use case diagram* dan *use case* skenario dari sistem DBA ticketing maupun sistem pintu air.

4.1 Waktu, Sumber Daya Manusia dan Biaya pengembangan milik perusahaan

Waktu yang diperlukan untuk pengembangan sistem DBA ticketing berdasarkan data milik perushaan adalah 4 bulan, sedangkan untuk sistem pintu air selama 4 bulan. Sumber daya manusia yang yang digunakan ketika mengembangkan sistem DBA ticketing adalah sebanyak 3 orang sedangkan sumber daya manusia yang digunakan ketika mengembangkan sistem pintu air sebanyak 2 orang. Biaya pengembangan sistem DBA ticketing adalah sebesar Rp 25.000.000 sedangkan biaya pengembangan sistem pintu air adalah sebesar Rp 35.000.000.

4.2 Sistem DBA *ticketing*

4.2.1 *Use case diagram* sistem DBA *ticketing*

Gambar 4.1 berikut merupakan *use case diagram* sistem DBA *ticketing.* Sistem divisualisasikan ke dalam *use case diagram* sesuai dengan hasil wawancara terhadap CV. Profile Image Studio. Langkah berikutnya adalah mengidentifikasi aktor untuk dikelompokkan berdasarkan karakteristik dari masing-masing aktor, lalu untuk menentukan tujuan aktor saat aktor menjalankan fungsionalitas pada sistem.

Gambar 4.1 Use Case Diagram DBA Ticketing

Gambar 4.1 merupakan gambar *use case diagram* sistem DBA *ticketing* yang mana terdiri dari 4 aktor dan 12 *use case*.

Tabel 4.1 Deskripsi Aktor pada sistem DBAticketing

Nama Aktor	Deskripsi
Guest	Aktor yang diperankan oleh orang yang dapat mengakses halaman awal sistem DBA *ticketing* secara langsung. Aktor dapat menggunakan sistemi sesuai dengan hak akses.
Agen	Aktor yang diperankan oleh pihak yang bekerjasama dengan pihak pembuat film untuk membantu menjual tiket, yang akan menggunakan sistem untuk proses pemesanan tiket dan mengatur deposit.
Admin OTS	Aktor yang diperankan oleh pegawai yang ditunjuk pihak pembuat film yang mana dapat menggunakan fungsi di dalam sistem tersebut seperti pemesanan tiket dan mengatur deposit.

Administrator	Aktor yang diperankan oleh pegawai yang ditunjuk menjadi admin oleh pihak pembuat film yang mana dapat menggunakan seluruh fungsi di dalam sistem tersebut seperti mengatur laporan, mengatur deposit, mengatur data agen dan mengatur jadwal.

Tabel 4.1 menjelaskan deskripsi dari masing-masing aktor yang ada pada *use case diagram* sistem DBA *ticketing*.

Tabel 4.2 Deskripsi *use case diagram sistem* DBAticketing

Nama Aktor	Kode	Nama Use Case	Deskripsi *Use Case*
Guest	UC-01	Login	*Use Case login* menjelaskan bagaimana aktor *guest* bisa masuk kedalam sistem untuk mengakses informasi sesuai hak akses.
	UC-02	Memesan Tiket	*Use Case* ini menjelaskan bagaimana proses aktor pengguna ketika melakukan pemesanan tiket.
	UC-03	Memeriksa status tiket	*Use Case* ini menjelaskan bagaimana proses aktor pengguna untuk memeriksa status tiket yang telah dipesan.
Agen	UC-04	Menambah Deposit	*Use Case* ini menjelaskan bagaimana aktor dapat menambahkan sejumlah deposit uang.
	UC-05	Melihat Laporan	*Use case* ini menjelaskan proses untuk melihat laporan penjualan tiket.
Admin OTS	UC-06	Memverifikasi pemesanan tiket	*Use case* ini menjelaskan mengenai proses untuk memverifikasi tiket yang sudah dipesan.
Administrator	UC-07	Menambahkan data agen	*Use Case* tersebut menjelaskan proses penambahan data agen ke sistem.
	UC-08	Melihat data agen	*Use case* ini menjelaskan bagaimana proses aktor untuk melihat informasi terkait data agen yang ada.
	UC-09	Mengubah data agen	*Use case* ini menjelaskan bagaimana proses aktor mengubah informasi pada data agen.
	UC-10	Menghapus data agen	*Use case* ini menjelaskan bagaimana proses aktor menghapus data agen yang sudah ada sebelumnya.

Nama Aktor	Kode	Nama Use Case	Deskripsi *Use Case*
	UC-11	Menambahkan jadwal	*Use Case* ini menjelaskan mengenai bagaimana proses penambahan jadwal penanyangan film oleh aktor.
	UC-12	Mengubah jadwal	*Use Case* ini menjelaskan mengenai proses aktor mengubah jadwal penanyangan film.

Tabel 4.2 menjelaskan deksripsi tiap *use case* yang ada pada *use case diagram* sistem DBA *ticketing* dan memberikan kode kepada masing-masing *use case.*

4.2.2 Use case skenario sistem DBA ticketing

Pada use case skenario akan dijelaskan lebih rinci mengenai aktifitas di dalam sebuah *use case.* Use case skenario tersusun dari deskripsi mengenai use case kemudian bagaimana keadaan sebelum *use case* dijalankan, keadaan aktor dan setelah use *case* dijalankan, dan urutan langkah yang harus dilewati oleh aktor dalam menyelesaikan suatu use case.

4.2.2.1 Skenario *use case* Login

Tabel 4.3 *Skenario* use case login

Brief Description	*Use Case login* menjelaskan bagaimana aktor pengguna bisa masuk kedalam sistem untuk mengakses informasi sesuai hak akses.
Aktor	*Guest*
Pre condition	• Aktor belum masuk ke dalam sistem. • *Device* tersambung ke internet. • Aktor telah berada di halaman login sistem.
Post condition	Aktor masuk ke dalam sistem pada halaman utama.
Basic Flow	1. *Use case* mulai saat aktor pengguna mengisi *form login* berupa nama dan sandi. 2. Aktor *guest* mengirim nama dan kata sandi ke dalam sistem. 3. Sistem melakukan identifikasi terhadap pengguna. 4. Sistem menampilkan hak akses sesuai hak akses pengguna. 5. *Use case* selesai
Alternative Flow	AF1 : Kondisi adanya kesalahan dalam proses identifikasi pengguna Jika sistem tidak berhasil dalam proses identifikasi terhadap pengguna maka sistem akan mengeluarkan pesan bahwa telah terjadi kesalahan pada pengisian data di *form login,* kemudian *use case* selesai.

Tabel 4.3 menjelaskan mengenai *use case* skenario dari *use case login* pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor berhasil masuk kedalam sistem.

4.2.2.2 Skenario *use case* memesan tiket

Tabel 4.4 *Skenario* use case memesan tiket

Brief Description	*Use Case* ini menjelaskan bagaimana proses aktor pengguna ketika melakukan pemesanan tiket.
Actor	*Guest*
Pre condition	• Aktor telah berada di halaman awal.
Post condition	Aktor telah berhasil mendapatkan kode pemesanan tiket.
Basic Flow	1. *Use case* mulai saat aktor memilih memesan tiket 2. Sistem menampilkan jadwal tayang. 3. Aktor memilih jadwal tayang. 4. Sistem menampilkan *form* pengisian identitas pemesan. 5. Aktor mengisi identitas pada *form*. 6. Sistem menampilkan form jumlah tiket yang akan dibeli. 7. Aktor mengisi jumlah tiket yang akan dibeli. 8. Sistem menampilkan pilihan nomor rekening yang akan digunakan untuk membayar. 9. Aktor memilih nomor rekening. 10. Sistem memproses pemesanan tiket oleh aktor 11. Sistem mengeluarkan pesan bahwa proses pemesanan telah berhasil. 12. Sistem menampilkan kode pemesanan tiket. 13. *Use case* selesai.
Alternative Flow	AF1 : Kondisi kegagalan dalam proses pemesanan tiket Jika sistem gagal dalam memproses pemesanan tiket maka sistem akan menampilkan pesan bahwa proses pemesanan tiket gagal dilakukan, kemudian *use case* selesai.

Tabel 4.4 menjelaskan mengenai *use case* skenario dari *use case* memesan tiket pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor berhasil mendapatkan kode pemesanan tiket tanpa masuk kedalam sistem.

Tabel 4.5 S*kenario* use case memesan tiket

Brief Description	*Use Case* ini menjelaskan bagaimana proses aktor pengguna ketika melakukan pemesanan tiket.
Actor	Agen, admin OTS dan administrator

Pre condition	• Aktor telah berada di halaman awal. • Aktor telah berhasil masuk ke dalam sistem.
Post condition	Aktor telah berhasil mendapatkan kode pemesanan tiket.
Basic Flow	1. *Use case* mulai saat aktor memilih untuk memesan tiket 2. Sistem menampilkan jadwal tayang. 3. Aktor memilih jadwal tayang. 4. Sistem menampilkan *form* pengisian identitas pemesan. 5. Aktor mengisi identitas pada *form*. 6. Sistem menampilkan form jumlah tiket yang akan dibeli. 7. Aktor mengisi jumlah tiket yang akan dibeli. 8. Sistem menampilkan pilihan nomor rekening yang akan digunakan untuk membayar. 9. Aktor memilih nomor rekening. 10. Sistem memproses pemesanan tiket oleh actor 11. Sistem mengeluarkan pesan bahwa prose pemesanan telah berhasil. 12. Sistem menampilkan kode pemesanan tiket. 13. *Use case* selesai.
Alternative Flow	AF1 : Kondisi kegagalan dalam proses pemesanan tiket Jika sistem gagal dalam memproses pemesanan tiket maka sistem akan mengeluarkan pesan bahwa proses pemesanan tiket, *use case* selesai.

Tabel 4.5 menjelaskan mengenai *use case* skenario dari *use case* memesan tiket pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor berhasil mendapatkan kode pemesanan tiket tanpa masuk kedalam sistem.

4.2.2.3 Skenario *use case* memeriksa status tiket

Tabel 4.6 Skenario *use case* memeriksa status tiket

Brief Description	*Use Case* ini menjelaskan bagaimana proses aktor pengguna untuk memeriksa status tiket yang telah dipesan sebelumnya.
Actor	Guest
Pre condition	• Aktor telah berada di halaman awal.
Post condition	Aktor telah berhasil mendapatkan informasi status tiket yang telah dipesan.
Basic Flow	1. *Use case* mulai saat aktor pengguna memilih untuk cek pembayaran 2. Sistem menampilkan form yang berisi kode tiket dan nomor *handphone* 3. Aktor mengisi form tersebut.

	4. Aktor mengirim kode dan nomor *handphone* ke dalam sistem. 5. Sistem akan mengidentifikasi kode dan nomor *handphone* tersebut. 6. Sistem mengeluarkan status tiket berdasarkan kode dan nomor *handphone* tersebut. 7. *Use case* selesai.
Alternative Flow	AF1 : Kondisi kegagalan dalam kesalahan pengisian *form* status tiket Jika sistem gagal dalam mengidentifikasi kode dan nomor *handphone* maka sistem akan menampilkan pesan masukan kode dan nomor *handphone* dengan benar, lalu *use case* selesai.

Tabel 4.6 menjelaskan mengenai *use case* skenario dari *use case* memeriksa status tiket pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah Aktor telah berhasil mendapatkan informasi status tiket yang telah dipesan.

Tabel 4.7 Skenario *use case* memeriksa status tiket

Brief Description	*Use Case* ini menjelaskan bagaimana proses aktor pengguna untuk memeriksa status tiket yang telah dipesan sebelumnya.
Actor	Agen, Admin OTS dan administrator
Pre condition	• Aktor telah berada di halaman awal. • Aktor telah berhasil masuk ke dalam sistem.
Post condition	Aktor telah berhasil mendapatkan informasi status tiket yang telah dipesan.
Basic Flow	1. *Use case* mulai saat aktor pengguna memilih untuk cek pembayaran 2. Sistem menampilkan form yang berisi kode tiket dan nomor *handphone* 3. Aktor mengisi form tersebut. 4. Aktor mengirim kode dan nomor *handphone* ke dalam sistem. 5. Sistem akan mengidentifikasi kode dan nomor *handphone* tersebut. 6. Sistem menampilkan status tiket berdasarkan kode dan nomor *handphone* tersebut. 7. *Use case* selesai.
Alternative Flow	AF1 : Kondisi kegagalan dalam kesalahan pengisian *form* status tiket Jika sistem gagal dalam mengidentifikasi kode dan nomor *handphone* maka sistem akan menampilkan pesan masukan kode dan nomor *handphone* dengan benar, lalu *use case* selesai.

Tabel 4.7 menjelaskan mengenai *use case* skenario dari *use case* memeriksa status tiket pada sistem DBA *ticketing* yang mana *post condition* dari

use case tersebut adalah Aktor telah berhasil mendapatkan informasi status tiket yang telah dipesan.

4.2.2.4 Skenario *use case* melihat laporan penjualan

Tabel 4.8 Skenario *use case* melihat laporan penjualan

Brief Description	*Use Case* ini menjelaskan bagaimana aktor dapat melihat hasil penjualan tiket yang telah berhasil dijual.
Actor	Agen, admin OTS dan administrator
Pre condition	• Aktor telah berhasil masuk ke dalam sistem.
Post condiiton	Aktor berhasil melihat laporan hasil penjualan tiket.
Basic Flow	1. *Use case* mulai saat aktor memilih melihat laporan penjualan. 2. Sistem memuat data pada laporan penjualan. 3. Sistem menampilkan data laporan penjualan. 4. Aktor memilih untuk mencari data pada laporan. 5. Aktor memasukkan kata kunci sesuai data yang dicari. 6. Sistem memuat data yang dicari. 7. Sistem menampilkan data yang dicari pada laporan. 8. *Use case* selesai.
Alternative Flow	AF1 : Kondisi ketika data yang dicari tidak ditemukan. Ketika dalam mencari data pada laporan tidak ditemukan maka tidak ada data yang akan ditampilkan oleh sistem, *use case* selesai.

Tabel 4.8 menjelaskan mengenai *use case* skenario dari *use case* melihat laporan penjualan pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah Aktor berhasil melihat laporan hasil penjualan tiket.

4.2.2.5 Skenario *use case* menambah deposit

Tabel 4.9 Skenario *use case* menambah deposit

Brief Description	*Use Case* menjelaskan cara aktor dapat menambahkan sejumlah deposit uang.
Actor	Agen, Admin OTS dan administrator
Pre condition	• Aktor telah berhasil masuk ke dalam sistem.
Post condition	Aktor berhasil menghapus data laporan penjualan.
Basic Flow	1. *Use case* mulai saat aktor memilih melihat data deposit. 2. Sistem menampilkan data deposit. 3. Aktor memilih menambahkan deposit. 4. Sistem menampilkan *form* penambahan deposit. 5. Aktor memasukkan jumlah nominal dan memilih rekening.

	6. Aktor memilih untuk menyimpan deposit. 7. Sistem memproses penyimpanan data deposit. 8. Sistem mengeluarkan pesan bahwa deposit berhasil dilakukan. 9. *Use case* selesai.
Alternative Flow	AF1 : Kondisi ketika terjadi kesalahan pengisian form untuk menambah deposit. Jika dalam proses penyimpanan data deposit terjadi kegagalan karena pengisian form yang tidak sesuai maka sistem akan mengeluarkan pesan jika proses penambahan deposit tidak berhasil dilakukan, *use case* selesai.

Tabel 4.9 menjelaskan mengenai *use case* skenario dari *use case* menambah deposit pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor berhasil menghapus data laporan penjualan.

4.2.2.6 Skenario *use case* memverifikasi pemesanan tiket

Tabel 4.10 Skenario *use case* memverifikasi pemesanan tiket

Brief Description	*Use Case* ini menjelaskan bagaimana aktor dapat memverifikasi pemesanan tiket.
Actor	Admin OTS dan administrator
Pre condition	• Aktor telah berhasil masuk ke dalam sistem.
Post condition	Aktor berhasil memverifikasi pemesanan tiket.
Basic Flow	1. *Use case* mulai saat aktor memilih memverifikasi tiket. 2. Sistem memuat data tiket yang dipesan. 3. Sistem menampilkan data tiket yang dipesan. 4. Aktor memilih data tiket untuk diverifikasi. 5. Sistem mengeluarkan pesan bahwa tiket sudah diverifikasi. 6. *Use case* selesai.
Alternative Flow	AF1 : Kondisi ketika pembayaran belum dilakukan. Jika dalam proses verifikasi gagal karena pembayaran belum dilakukan maka sistem akan mengeluarkan pesan bahwa proses verifikasi tiket tidak berhasil dilakukan, *use case* selesai.

Tabel 4.10 menjelaskan mengenai *use case* skenario dari *use case* memverifikasi pemesanan tiket pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor berhasil memverifikasi pemesanan tiket.

4.2.2.7 Skenario *use case* menambah data agen

Tabel 4.11 Skenario *use case* menambah data agen

Brief Description	*Use Case* menjelaskan proses penambahan data agen ke dalam sistem oleh aktor.
Actor	Administrator
Pre condition	• Aktor telah masuk ke dalam sistem.
Post condition	Aktor sukses menambahkan data agen ke dalam sistem
Basic Flow	1. *Use case* mulai saat aktor pengguna memilih untuk melihat data agen. 2. Sistem memuat data agen. 3. Sistem menampilkan data agen. 4. Aktor memilih untuk menambahkan data agen baru. 5. Sistem mengeluarkan *form* penambahan agen baru. 6. Aktor mengisi *form* pendaftaran agen baru. 7. Aktor memilih menyimpan data agen baru. 8. Sistem memproses penyimpanan data agen baru. 9. Sistem mengeluarkan pesan jika data agen baru berhasil disimpan. 10. *Use case* selesai.
Alternative Flow	AF1 : Kondisi kesalahan pengisian form. Jika sistem gagal dalam proses penyimpanan data agen baru karena kesalahan pengisian *form*, sistem akan menampikan pesan jika proses penambahan agen baru gagal, *use case* selesai.

Tabel 4.11 menjelaskan mengenai *use case* skenario dari *use case* menambah data agen pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor sukses menambahkan data agen ke dalam sistem.

4.2.2.8 Skenario *use case* melihat data agen

Tabel 4.12 Skenario *use case* melihat data agen

Brief Description	*Use Case* ini menjelaskan proses melihat data agen yang ada dalam sistem oleh aktor.
Actor	Administrator
Pre condition	• Aktor telah berhasil masuk ke dalam sistem.
Post condition	Aktor dapat melihat daftar agen yang ada.
Basic Flow	1. *Use case* mulai saat aktor pengguna memilih melihat data agen.

	2. Sistem memuat data agen. 3. Sistem tampilkan data agen. 4. *Use case* selesai.
Alternative Flow	AF1 : Kondisi kegagalan ketika data agen tidak ada. Jika sistem gagal dalam proses mencaridan menampilkan data agen karena data tidak ada, maka sistem akan menampikan pesan bahwa data tidak ditemukan.

Tabel 4.12 menjelaskan mengenai *use case* skenario dari *use case* melihat data agen pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor dapat melihat daftar agen yang ada.

4.2.2.9 Skenario *use case* mengubah data agen

Tabel 4.13 Skenario *use case* mengubah data agen

Brief Description	*Use case* menjelaskan bagaimana proses aktor mengubah informasi pada data agen.
Actor	Administrator
Pre condition	• Aktor telah masuk ke dalam sistem. • Data agen tersimpan di dalam sistem
Post condition	Aktor sukses megubah data agen di dalam sistem.
Basic Flow	1. *Use case* dimulai ketika aktor pengguna memilih untuk melihat data agen. 2. Sistem memuat data agen. 3. Sistem menampilkan data agen. 4. Aktor memilih salah satu agen untuk diubah datanya. 5. Sistem mengeluarkan *form* berisi data agen. 6. Aktor merubah data agen yang ingin diubah. 7. Aktor memilih memperbarui data agen. 8. Sistem memproses pembaruan data agen. 9. Sistem mengeluarkan pesan jika data agen berhasil diperbarui. 10. *Use case* selesai.
Alternative Flow	AF1 : Kondisi kesalahan pengisian *form* data agen. Jika sistem gagal dalam proses pembaruan data agen baru karena kesalahan pengisian *form*, maka sistem menampikan pesan jika proses pembaruan agen baru gagal, *use case* selesai.

Tabel 4.12 menjelaskan mengenai *use case* skenario dari *use case* mengubah data agen pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor sukses megubah data agen di dalam sistem.

4.2.2.10 Skenario *use case* menghapus data agen

Tabel 4.14 Skenario *use case* menghapus data agen

Brief Description	*Use case* menjelaskan bagaimana proses aktor menghapus data agen yang sudah ada sebelumnya.
Actor	Administrator
Pre condition	• Aktor berhasil masuk ke dalam sistem. • Data agen tersimpan dalam sistem
Post condition	Aktor sukses menghapus data agen di dalam sistem.
Basic Flow	1. *Use case* mulai saat aktor memilih untuk melihat data para agen. 2. Sistem memuat data para agen. 3. Sistem menampilkan data para agen. 4. Aktor memilih agen yang akan dihapus. 5. Sistem memproses penghapusan agen. 6. Sistem menampilkan pesan bahwa data agen berhasil dihapus. 7. *Use case* selesai.
Alternative Flow	AF1 : Kondisi ketika data agen tidak berhasil dihapus. Jika dalam proses penghapusan data agen sistem mengalami kegagalan maka sistem akan menampilkan pesan bahwa menghapus data laporan tidak berhasil, *use case* berkahir.

Tabel 4.14 menjelaskan mengenai *use case* skenario dari *use case* menghapus data agen pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor sukses menghapus data agen di dalam sistem.

4.2.2.11 Skenario *use case* menambah jadwal

Tabel 4.15 Skenario *use case* menambah jadwal

Brief Description	*Use Case* menjelaskan proses penambahan jadwal ke dalam sistem.
Actor	Administrator
Pre condition	• Aktor telah masuk ke dalam sistem.
Post condition	Aktor berhasil menambahkan jadwal ke dalam sistem.
Basic Flow	1. *Use case* mulai saat aktor memilih melihat jadwal tayang. 2. Sistem memuat jadwal tayang. 3. Sistem menampilkan jadwal tayang. 4. Aktor memilih menambah jadwal baru. 5. Sistem menampilkan *form* penambahan jadwal baru. 6. Aktor mengisi *form* jadwal baru.

	7. Aktor memilih menyimpan jadwal baru. 8. Sistem memproses penyimpanan jadwal tayang baru. 9. Sistem mengeluarkan pesan jika jadwal tayang berhasil ditambahkan. 10. *Use case* selesai.
Alternative Flow	AF1 : Kondisi kesalahan pengisian *form* jadwal tayang. Jika sistem gagal dalam proses penyimpanan jadwal tayang baru karena kesalahan pengisian *form*, sistem akan menampikan pesan jika proses penambahan jadwal baru gagal, *use case* selesai.

Tabel 4.15 menjelaskan mengenai *use case* skenario dari *use case* menambah jadwal pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor berhasil menambahkan jadwal ke dalam sistem.

4.2.2.12 Skenario *use case* mengubah jadwal

Tabel 4.16 Skenario *use case* mengubah jadwal

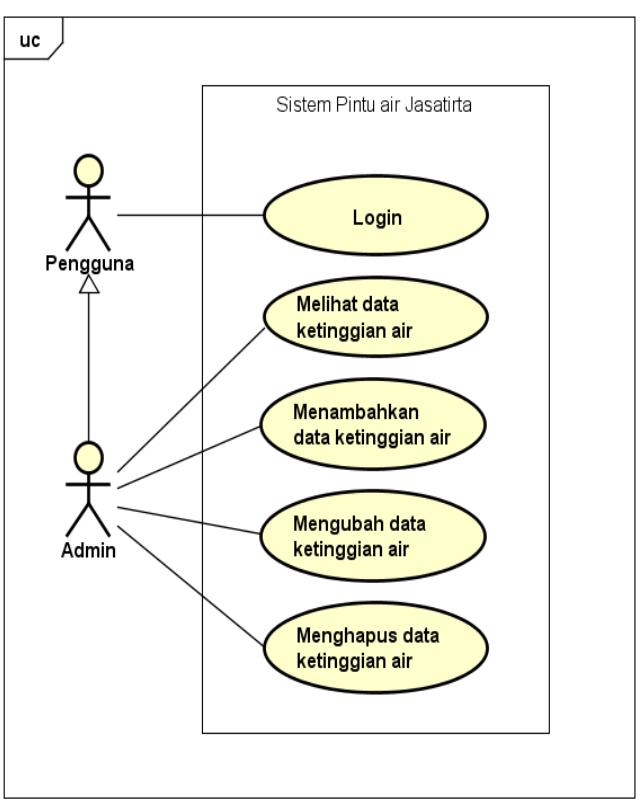
Brief Description	*Use Case* menjelaskan mengenai proses aktor mengubah jadwal penanyangan film.
Actor	Administrator
Pre condition	• Aktor telah berada di dalam sistem. • Data jadwal telah tersimpan di dalam sistem
Post condition	Aktor sukses megubah jadwal tayang di sistem.
Basic Flow	1. *Use case* mulai saat aktor pengguna memilih untuk melihat jadwal tayang. 2. Sistem memuat jadwal tayang. 3. Sistem menampilkan jadwal tayang. 4. Aktor memilih jadwal yang ingin diubah. 5. Sistem menampilkan *form* berisi data jadwal yang akan diubah. 6. Aktor mengubah data jadwal yang ingin diubah. 7. Aktor memilih memperbarui data jadwal tayang. 8. Sistem memproses pembaruan data jadwal tayang. 9. Sistem mengeluarkan pesan bahwa jadwal tayang berhasil diperbarui. 10. *Use case* selesai.
Alternative Flow	AF1 : Menangani kesalahan pengisian *form* untuk jadwal baru. Jika sistem gagal dalam proses pembaruan jadwal tayang baru karena kesalahan pengisian *form*, maka sistem akan menampikan pesan bahwa proses pembaruan jadwal baru gagal, lalu *use case* selesai.

Tabel 4.16 menjelaskan mengenai *use case* skenario dari *use case* mengubah jadwal pada sistem DBA *ticketing* yang mana *post condition* dari use case tersebut adalah aktor sukses megubah jadwal tayang di sistem.

4.3 Sistem pintu air

4.3.1 *Use case diagram* sistem pintu air

Gambar 4.2 berikut merupakan *use case diagram* sistem pintu air. Sistem divisualisasikan ke dalam *use case diagram* sesuai dengan hasil wawancara terhadap CV. Profile Image Studio. Langkah berikutnya adalah mengidentifikasi aktor untuk dikelompokkan berdasarkan karakteristik dari masing-masing aktor, lalu untuk menentukan tujuan aktor saat aktor menjalankan fungsionalitas pada sistem.



Gambar 4.2 use case

Tabel 4.17 Deskripsi aktor

Nama Aktor	Deskripsi
Pengguna	Aktor pengguna diperankan oleh orang yang dapat mengakses sistem pintu air secara langsung. Aktor dapat menggunakan sistem sesuai dengan hak aksesnya.
Administrator	Aktor yang diperankan oleh pegawai yang ditunjuk menjadi admin oleh pihak jasatirta yang mana dapat menggunakan

	seluruh fungsi di dalam sistem tersebut seperti menambahkan data ketinggian air, mengubah data ketinggian air, serta menghapus data ketinggian air.

Tabel 4.17 menjelaskan mengenai deskripsi dari setiap aktor yang terdapat pada *use case diagram* sistem pintu air.

Tabel 4.18 Deskripsi Use case

Nama Aktor	Kode	Nama Use Case	Deskripsi *Use Case*
Pengguna	UC-01	Login	*Use Case* menjelaskan bagaimana aktor pengguna bisa masuk kedalam sistem untuk mengakses informasi sesuai hak akses.
Administrator	UC-04	Melihat data ketinggian air	*Use Case* menjelaskan bagaimana aktor dapat melihat data dari ketinggian air yang ada.
	UC-05	Menambah data ketinggian air	*Use Case* menjelaskan bagaimana aktor dapat menambah data ketinggiar air ke dalam sistem.
	UC-06	Mengubah data ketinggian air	*Use Case* menjelaskan bagaimana aktor dapat memperbarui data ketinggiar air yang sudah ada.
	UC-07	Menghapus data ketinggian air	*Use Case* menjelaskan bagaimana aktor dapat menghapus data ketinggian air yang ada.

Tabel 4.18 menjelaskan mengenai deksripsi setiap *use case* yang terdapat pada *use case diagram* sistem Pintu air serta memberikan kode kepada masing-masing *use case.*

4.3.2 *Use case* skenario sistem pintu air

Pada use case skenario akan dijelaskan lebih rinci mengenai aktifitas di dalam sebuah *use case.* Use case skenario tersusun dari deskripsi mengenai use case kemudian bagaimana kondisi sebelum *use case* dijalankan, kondisi aktor dan setelah use *case* dijalankan, dan urutan langkah yang harus dilakukan oleh aktor dalam menyelesaikan suatu *use case*.

4.3.2.1 Skenario *Use Case Login*

Tabel 4.19 Skenario use case login

Brief Description	*Use Case* ini menjelaskan bagaimana aktor pengguna bisa masuk kedalam sistem untuk mengakses informasi sesuai hak akses.

Actor	Pengguna
Pre condition	• Aktor pengguna tidak masuk ke dalam sistem. • *Device* tersambungan ke internet. • Aktor Agen telah berada di halaman login sistem.
Post condition	Aktor telah masuk ke dalam sistem pada halaman utama.
Basic Flow	1. *Use case* mulai saat aktor pengguna mengisi *form login* berupa nama dan kata sandi. 2. Aktor pengguna memasukkan nama dan sandi ke dalam sistem. 3. Sistem melakukan identifikasi terhadap pengguna. 4. Aktor masuk dihalaman utama sistem. 5. *Use case* selesai
Alternative Flow	AF1 : Menangani adanya kesalahan dalam proses identifikasi pengguna Jika sistem gagal dalam proses identifikasi terhadap aktor maka sistem akan mengeluarkan pesan jika telah terjadi kesalahan pada pengisian data di *form login, use case* selesai.

Tabel 4.19 menjelaskan mengenai *use case* skenario dari *use case login* pada sistem Pintu Air yang mana *post condition* dari use case tersebut adalah aktor berhasil masuk kedalam sistem.

4.3.2.2 Skenario *use case* melihat data ketinggian air

Tabel 4.20 Skenario use case melihat data ketinggian air

Brief Description	*Use Case* menjelaskan bagaimana aktor dapat melihat data ketinggian air yang ada di dalam sistem.
Actor	Administrator
Pre condition	• Aktor telah masuk ke dalam sistem.
Post condition	Aktor sukses melihat data ketinggian air yang ada.
Basic Flow	1. *Use case* mulai saat aktor memilih untuk melihat data ketinggian air. 2. Sistem memuat data ketinggian air. 3. Sistem menampilkan data ketinggian air. 4. Aktor memilih untuk mencari data sesuai kata kunci pada data ketinggian air. 5. Aktor memasukkan kata kunci sesuai yang ingin dicari. 6. Sistem memuat data yang dicari. 7. Sistem mengeluarkan data yang dicari pada laporan. 8. *Use case* selesai.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

Alternative Flow	AF1 : Ketika kondisi data yang dicari tidak ditemukan. Jika dalam mencari data tidak ditemukan maka tidak ada yang akan ditampilkan oleh sistem, *use case* selesai.

Tabel 4.20 menjelaskan mengenai *use case* skenario dari *use case* meihat ketinggain air pada sistem Pintu Air yang mana *post condition* dari use case tersebut adalah aktor berhasil masuk kedalam sistem.

4.3.2.3 Skenario *use case* menambah data ketinggian air

Tabel 4.21 Skenario use case menambah data ketinggian air

Brief Description	*Use Case* menjelaskan proses penambahan data ketinggian air ke dalam sistem.
Actor	Administrator
Pre condition	• Aktor telah berhasil masuk ke dalam sistem.
Post condition	Aktor berhasil menambahkan data ketinggian air ke dalam sistem.
Basic Flow	1. *Use case* mulai saat aktor memilih untuk melihat data ketinggian air. 2. Sistem memuat data ketinggian air. 3. Sistem menampilkan data ketinggian air. 4. Aktor memilih untuk menambah data ketinggian air. 5. Sistem menampilkan *form* penambahan data ketinggian air baru. 6. Aktor mengisi *form* penambahan data ketinggian air baru. 7. Aktor memilih menyimpan data ketinggian air baru. 8. Sistem melakukan proses penyimpanan data ketinggian air baru. 9. Sistem mengeluarkan pesan bahwa data ketinggian air baru berhasil disimpan. 10. *Use case* selesai.
Alternative Flow	AF1 : Menangani kegagalan penambahan data ketinggian air baru. Jika sistem gagal dalam proses penyimpanan data ketinggian air baru, maka sistem akan menampikan pesan jika proses penambahan data ketinggian air baru gagal, lalu *use case* selesai.

Tabel 4.21 menjelaskan mengenai *use case* skenario dari *use case* menambah data ketinggain air pada sistem Pintu Air yang mana *post condition* dari use case tersebut adalah aktor berhasil menambahkan data ketinggian air ke dalam sistem.

4.3.2.4 Skenario *use case* mengubah data ketinggian air

Tabel 4.22 Skenario use case megubah data ketinggian air

Brief Description	*Use Case* menjelaskan bagaimana aktor dapat mengubah data ketinggian air.
Actor	Administrator
Precondition	• Aktor telah masuk ke dalam sistem. • Data ketinggian air tersimpan didalam sistem.
Postcondition	Aktor sukses mengubah data ketinggian air yang ada.
Basic Flow	1. *Use case* mulai saat aktor memilih untuk melihat data ketinggian air 2. Sistem memuat data memuat data ketinggian air. 3. Sistem menampilkan data ketinggian air. 4. Aktor memilih data ketinggian air yang akan diubah. 5. Sistem menampilkan *form* untuk mengubah data. 6. Aktor mengubah data yang dipilih. 7. Aktor memilih untuk memperbarui data ketinggian air. 8. Sistem melakukan proses perubahan data ketinggian air. 9. Sistem mengeluarkan pesan jika data telah berhasil diperbarui. 10. *Use case* selesai.
Alternative Flow	AF1 : Menangani kondisi ketika data ketinggian air gagal diperbarui. Jika dalam proses perubahan data laporan sistem mengalami kegagalan sistem akan mengeluarkan pesan jika proses pembaruan data tidak berhasil, *use case* berkahir.

Tabel 4.22 menjelaskan mengenai *use case* skenario dari *use case* mengubah data ketinggain air pada sistem Pintu Air yang mana *post condition* dari use case tersebut adalah aktor sukses mengubah data ketinggian air yang ada.

4.3.2.5 Skenario *use case* menghapus data ketinggian air

Tabel 4.23 Skenario menghapus data ketinggian air

Brief Description	*Use Case* menjelaskan bagaimana aktor dapat menghapus data ketinggian air yang ada di dalam sistem.
Actor	Administrator
Pre condition	• Aktor telah masuk ke dalam sistem. • Data ketinggian air tersimpan dalam sistem.
Post condition	Aktor sukses menghapus data ketinggian air dalam sistem.
Basic Flow	1. *Use case* mulai saat aktor memilih untuk melihat data ketinggian air. 2. Sistem memuat data ketinggian air.

	3. Sistem menampilkan data ketinggian air. 4. Aktor memilih data ketinggian air yang akan dihapus. 5. Sistem melakukan proses penghapusan data ketinggian air. 6. Sistem mengeluarkan pesan jika data ketinggian air telah berhasil dihapus. 7. *Use case* selesai.
Alternative Flow	AF1 : Ketika kondisi data ketinggian air gagal dihapus. Jika dalam proses penghapusan data ketinggian air sistem mengalami kegagalan sistem akan mengeluarkan pesan bahwa menghapus data ketinggian air tidak berhasil, *use case* berkahir.

Tabel 4.23 menjelaskan mengenai *use case* skenario dari *use case* menghapus data ketinggain air pada sistem Pintu Air yang mana *post condition* dari use case tersebut adalah Aktor sukses menghapus data ketinggian air dalam sistem.

BAB 5 PEMBAHASAN

5.1 Menghitung *Unadjusted Use Case Point* Sistem DBA *ticketing*

Untuk mendapatkan nilai *unadjusted use case point* didapatkan dari penjumlahan nilai *unadjusted use case weight* (UUCW) dengan nilai *unadjusted actor weight* (UAW) mengacu pada persamaan 2. Menghitung *unadjusted use case weight* dibutuhkan jumlah transaksi di setiap *use case*. Untuk menghitung *unadjusted actor weight* diperlukan informasi mengenai antarmuka yang digunakan aktor saat berinteraksi dengan sistem. Aktor akan dikategorikan dengan melihat bagaimana cara aktor berinteraksi dengan sistem sesuai pada table 5.1. Berikut adalah tabel penghitungan nilai *Unadjusted Actor Weight* (UAW) sistem DBA *ticketing*.

Tabel 5.1 *Unadjusted Actor Weight DBA ticketing*

No	Nama Aktor	Kategori Aktor	Bobot Aktor
1	Agen	Kompleks	3
2	Administrator	Kompleks	3
3	Admin OTS	Kompleks	3
Total *Unadjusted Actor Weight* (UAW)			9

Menurut tabel 5.1 aktor agen, administrator dan admin OTS masuk kedalam kategori kompleks dan memiliki nilai bobot 3 pada masing-masing aktor. Hal tersebut dikarenakan aktor agen, administrator dan admin OTS berinteraksi dengan sistem melalui *Graphical User Interface* (GUI). Total nilai *Unadjusted Actor Weight* (UAW) pada sistem DBA *ticketing* adalah **9**.

5.1.1 Transaksi Use Case

Transaksi dapat dihitung dengan cara menghitung jumlah langkah yang ada pada *basic flow* dan *alternative flow* yang ada didalam sebuah use case (Kristen Ribu, 2001). Terdapat tiga kategori yang berdasarkan jumlah dari transaksi yang terjadi dalam *use case.* Kategori tersebut yaitu sederhana, medium atau rata-rata, atau kompleks. Berikut merupakan transaksi yang ada pada sistem DBA *ticketing*.

1. Transaksi *use case* Login

Tabel 5.2 Tabel transaksi use case login

Basic flow : 1. Aktor *guest* mengisi *form login* berupa nama dan sandi. 2. Sistem memvalidasi data pada *form* login. 3. Sistem melakukan autentikasi terhadap aktor *guest*. 4. Sistem menampilkan halaman awal pengguna. *Alternative Flow* :

1. Menangani adanya kesalahan dalam proses identifikasi aktor *guest*
Jumlah Transaksi : 5

Berdasarkan tabel 5.2 jumlah transaksi yang terjadi pada *use case login* adalah 5 transaksi, hal tersebut dihitung berdasarkan 4 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case login*.

2. Transaksi use case memesan tiket

Tabel 5.3 Transaksi use case memesan tiket

Basic flow : 1. *Use case* dimulai ketika aktor pengguna memilih untuk memesan tiket 2. Sistem menampilkan jadwal tayang. 3. Aktor memilih jadwal tayang. 4. Sistem menampilkan *form* pengisian identitas pemesan. 5. Aktor mengisi identitas pada *form*. 6. Sistem menampilkan form jumlah tiket yang akan dibeli. 7. Aktor mengisi jumlah tiket yang akan dibeli. 8. Sistem menampilkan pilihan nomor rekening yang akan digunakan untuk membayar. 9. Aktor memilih nomor rekening. 10. Sistem melakukan proses pemesanan tiket oleh actor 11. Sistem menampilkan pesan bahwa prose pemesanan telah berhasil. 12. Sistem menampilkan kode pemesanan tiket. *Alternative Flow* : 1. Menangani kegagalan dalam proses pemesanan tiket
Jumlah transaksi : 13

Berdasarkan tabel 5.3 jumlah transaksi yang terjadi pada *use case* memesan tiket adalah 13 transaksi, hal tersebut dihitung berdasarkan 12 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* memesan tiket.

3. Transaksi use case memeriksa status tiket

Tabel 5.4 Transaksi use case memeriksa status tiket

Basic flow : 1. *Use case* dimulai ketika aktor pengguna memilih untuk cek pembayaran 2. Sistem menampilkan form yang berisi kode tiket dan nomor *handphone* 3. Aktor mengisi form tersebut. 4. Aktor mengirim kode dan nomor *handphone* ke dalam sistem. 5. Sistem akan mengidentifikasi kode dan nomor *handphone* tersebut.

6. Sistem menampilkan status tiket berdasarkan kode dan nomor *handphone* tersebut. *Alternative Flow* : 1. Menangani kegagalan dalam kesalahan pengisian *form* status tiket
Jumlah transaksi : 7

Berdasarkan tabel 5.4 jumlah transaksi yang terjadi pada *use case* memeriksa tiket adalah 7 transaksi, hal tersebut dihitung berdasarkan 6 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* memeriksa tiket.

4. Transaksi use case melihat laporan

Tabel 5.5 Transaksi use case melihat laporan

Basic flow: 1. *Use case* mulai saat aktor memilih untuk melihat laporan penjualan. 2. Sistem memuat data pada laporan penjualan. 3. Sistem menampilkan data laporan penjualan. 4. Aktor memilih untuk mencari data pada laporan. 5. Aktor memasukkan kata kunci. 6. Sistem memuat data yang dicari. 7. Sistem menampilkan data yang dicari pada laporan. *Alternative Flow* : 1. Menangani kondisi ketika data yang dicari tidak ditemukan.
Jumlah transaksi : 8

Berdasarkan tabel 5.5 jumlah transaksi yang terjadi pada *use case* melihat laporan adalah 8 transaksi, hal tersebut dihitung berdasarkan 7 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* melihat laporan.

5. Transaksi use case menambah deposit

Tabel 5.6 Transaksi use case menambah deposit

Basic flow: 1. *Use case* dimulai ketika aktor memilih untuk melihat data deposit. 2. Sistem menampilkan data deposit. 3. Aktor memeilih menambah deposit. 4. Sistem menampilkan *form* penambahan deposit. 5. Aktor memasukkan jumlah nominal dan memilih rekening. 6. Aktor memilih untuk menyimpan deposit. 7. Sistem melakukan proses penyimpanan data deposit. 8. Sistem mengeluarkan pesan bahwa deposit berhasil dilakukan.

repository.ub.ac.id
UNIVERSITAS BRAWIJAYA

Alternative Flow : 1. Menangani kondisi ketika terjadi kesalahan pengisian form untuk menambah deposit.
Jumlah transaksi : 9

Berdasarkan tabel 5.6 jumlah transaksi yang terjadi pada *use case* menambah deposit adalah 9 transaksi, hal tersebut dihitung berdasarkan 8 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* menambah deposit.

6. Transaksi use case memverifikasi pemesanan tiket

Tabel 5.7 Transaksi use case memverifikasi pemesanan tiket

Basic flow : 1. *Use case* dimulai ketika aktor memilih untuk memverifikasi tiket. 2. Sistem memuat data tiket yang dipesan. 3. Sistem menampilkan data tiket yang dipesan. 4. Aktor memilih data tiket untuk diverifikasi. 5. Sistem menampilkan pesan bahwa tiket sudah diverifikasi. *Alternative Flow* : 1. Menangani kondisi ketika pembayaran belum dilakukan.
Jumlah transaksi : 6

Berdasarkan tabel 5.7 jumlah transaksi yang terjadi pada *use case* memverifikasi pemesanan tiket adalah 6 transaksi, hal tersebut dihitung berdasarkan 5 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* memverifikasi pemesanan tiket.

7. Transaksi *use case* menambah data agen

Tabel 5.8 Transaksi use case menambah data agen

Basic flow: 1. *Use case* mulai saat aktor pengguna memilih untuk melihat data agen. 2. Sistem memuat data agen. 3. Sistem menampilkan data agen. 4. Aktor memilih untuk menambah data agen baru. 5. Sistem menampilkan *form* penambahan agen baru. 6. Aktor mengisi *form* pendaftaran agen baru. 7. Aktor memilih menyimpan data agen baru. 8. Sistem memproses penyimpanan data agen baru. 9. Sistem mengeluarkan pesan bahwa data agen baru berhasil disimpan. *Alternative Flow* : 1. Menangani kesalahan pengisian form.

Jumlah transaksi: 10

Berdasarkan tabel 5.8 jumlah transaksi yang terjadi pada *use case* menambah data agen adalah 10 transaksi, hal tersebut dihitung berdasarkan 9 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* menambah data agen.

8. Transaksi use case melihat data agen

Tabel 5.9 Transaksi *use case* melihat data agen

Basic flow: 1. *Use case* mulai saat aktor pengguna memilih untuk melihat data agen. 2. Sistem memuat data agen. 3. Sistem menampilkan data agen. 4. Aktor memilih untuk menambah data agen baru. 5. Sistem menampilkan *form* penambahan agen baru. 6. Aktor mengisi *form* pendaftaran agen baru. 7. Aktor memilih menyimpan data agen baru. 8. Sistem memproses penyimpanan data agen baru. 9. Sistem mengeluarkan pesan bahwa data agen baru berhasil disimpan. *Alternative Flow* : 1. Menangani kegagalan ketika data agen tidak ada.
Jumlah transaksi: 10

Berdasarkan tabel 5.9 jumlah transaksi yang terjadi pada *use case* melihat data agen adalah 10 transaksi, hal tersebut dihitung berdasarkan 9 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* melihat data agen.

9. Transaksi use case mengubah data agen

Tabel 5.10 Transaksi use case mengubah data agen

Basic flow: 1. *Use case* mulai saat aktor pengguna memilih untuk melihat data agen. 2. Sistem memuat data agen. 3. Sistem menampilkan data agen. 4. Aktor memilih salah satu agen untuk diubah datanya. 5. Sistem menampilkan *form* berisi data agen. 6. Aktor mengubah data agen yang ingin diubah. 7. Aktor memilih memperbarui data agen. 8. Sistem memproses pembaruan data agen. 9. Sistem mengeluarkan pesan bahwa data agen berhasil diperbarui. *Alternative Flow* :

1. Menangani kesalahan pengisian *form* data agen.
Jumlah transaksi: 10

Berdasarkan tabel 5.10 jumlah transaksi yang terjadi pada *use case* mengubah data agen adalah 10 transaksi, hal tersebut dihitung berdasarkan 9 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* mengubah data agen.

10. Transaksi use case menghapus data agen

Tabel 5.11 Transaksi use case menghapus data agen

Basic flow: 1. *Use case* mulai saat aktor memilih untuk melihat data para agen. 2. Sistem memuat data para agen. 3. Sistem menampilkan data para agen. 4. Aktor memilih agen yang akan dihapus. 5. Sistem memproses penghapusan agen. 6. Sistem mengeluarkan pesan bahwa data agen berhasil dihapus. *Alternative Flow* : 1. Menangani kondisi ketika data agen tidak berhasil dihapus.
Jumlah transaksi: 7

Berdasarkan tabel 5.11 jumlah transaksi yang terjadi pada *use case* menghapus data agen adalah 7 transaksi, hal tersebut dihitung berdasarkan 6 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* menghapus data agen.

11. Transaksi use case menambah jadwal

Tabel 5.12 Transaksi use case menambah jadwal

Basic flow: 1. *Use case* mulai saat aktor memilih untuk melihat jadwal tayang. 2. Sistem memuat jadwal tayang. 3. Sistem menampilkan jadwal tayang. 4. Aktor memilih untuk menambah jadwal baru. 5. Sistem menampilkan *form* penambahan jadwal baru. 6. Aktor mengisi *form* jadwal baru. 7. Aktor memilih menyimpan jadwal baru. 8. Sistem memproses penyimpanan jadwal tayang baru. 9. Sistem mengeluarkan pesan bahwa jadwal tayang berhasil ditambahkan. *Alternative Flow* : 1. Menangani kesalahan pengisian *form* jadwal tayang.
Jumlah transaksi: 10

Berdasarkan tabel 5.12 jumlah transaksi yang terjadi pada *use case* menambah jadwal adalah 10 transaksi, hal tersebut dihitung berdasarkan 9 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* menambah jadwal.

12. Transaksi use case mengubah jadwal

Tabel 5.13 Transaksi use case mengubah jadwal

Basic flow: 1. *Use case* mulai saat aktor pengguna memilih untuk melihat jadwal tayang. 2. Sistem memuat jadwal tayang. 3. Sistem menampilkan jadwal tayang. 4. Aktor memilih salah satu jadwal yang ingin diubah. 5. Sistem menampilkan *form* berisi data jadwal yang akan diubah. 6. Aktor mengubah data jadwal yang ingin diubah. 7. Aktor memilih memperbarui data jadwal tayang. 8. Sistem memproses pembaruan data jadwal tayang. 9. Sistem mengeluarkan pesan bahwa jadwal tayang berhasil diperbarui. *Alternative Flow* : 1. Menangani kesalahan pengisian *form* untuk jadwal baru.
Jumlah transaksi: 10

Berdasarkan tabel 5.13 jumlah transaksi yang terjadi pada *use case* mengubah jadwal adalah 10 transaksi, hal tersebut dihitung berdasarkan 9 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* mengubah jadwal.

Tabel 5.14 Penghitungan UUCW DBA *ticketing*

Jumlah transaksi pada *use case*	**Komplesitas**	**Jumlah *Use Case***	**Bobot**	**Bobot * Jumlah *Use case***
Kurang dari 4 transaksi	Sederhana	0	5	0
4 hingga 7 transaksi	Medium	4	7	28
Lebih dari 7 transaksi	Kompleks	8	15	120
Total UUCW				148

Menurut tabel 5.14 pada sistem DBA *ticketing* memiliki 4 *use case* yang memiliki jumlah transaksi antara 4 hingga 7 transaksi sehingga 4 *use case* tersebut masuk kedalam kategori medium, kemudian 8 *use case* memiliki jumlah transaksi lebih dari 7 sehingga masuk kedalam kategori kompleks. Nilai *Unadjusted Use Case Weight* (UUCW) pada sistem DBA *ticketing* didapat dengan *mengkalikan* bobot setiap tingkatan transaksi dengan banyaknya *use case* pada setiap tingkatan transaksi dan hasil nilai *Unadjusted Use Case Weight* (UUCW) adalah **148**.

Tabel 5.15 Penghitungan *Unadjusted Use Case Point* DBA *ticketing*

	Hasil
Unadjusted Use Case Weight (UUCW)	148
Unadjusted Actor Weight (UAW)	9
UUCW + UAW =	157

Dengan hasil perhitungan *unadjusted use case weight* (UUCW) dan *unadjusted actor weight* (UAW), didapatkan nilai *unadjusted use case point* (UUCP) pada sistem DBA *ticketing*. Pada Tabel 5.15 menjumlahkan nilai *unadjusted use case weight* dan *unadjusted actor weight* sehingga ditemukan bahwa nilai *unadjusted use case point* untuk sistem DBA ticketing adalah **157.**

5.1.2 Menghitung Technical Complexity Factor

Pada penelitian ini penulis memberikan nilai 3 terhadap *technical factor* karena sesuai dengan penelitian *(*Ani dan Basri, 2013) pada penelitian tersebut menjelaskan alasan utama mengapa nilai 3 diberikan pada *technical factor* adalah karena *use case* yang ada pada sistem kurang dari 50 *use case*. Hal tersebut berarti sistem yang dikembangkan tidak rumit. Jumlah *use case* pada sistem DBA *ticketing* adalah 12 *use case,* maka penulis menetapkan nilai 3 pada *technical factor*. Nilai 3 pada *technical factor* berarti faktor tersebut memiliki pengaruh yang biasa saja terhadap pengembangan sistem.

Tabel 5.16 Penilaian *technical factor* DBA *Ticketing*

Technical factor		**Nilai**	**Deskripsi**
1	Sistem Terdistribusi	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
2	Performa (respon time)	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
3	Efisiensi dari *user interface*	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
4	Kompleksitas proses	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
5	Penggunaan kembali kode program	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
6	Kemudahan Instalasi	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
7	Kemudahaan Pengoperasian	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
8	Dapat diaplikasikan diberbagai platform	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.

Technical factor		Nilai	Deskripsi
9	Mudah untuk dirubah	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
10	Konkurensi	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
11	Fitur Keamanan	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
12	Ketergantungan pada pihak ketiga	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
13	Perlunya Pelatihan Khusus	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.

Tabel 5.16 menjelaskan tentang deskripsi dari nilai yang diberikan terhadap masing-masing faktor pada *technical factor* sistem DBA *ticketing*.

Tabel 5.17 Penghitungan *technical factor*

Technical factor		Bobot	Nilai	Bobot * Nilai
1	Sistem Terdistribusi	2	3	6
2	Kinerja	1	3	3
3	Efisiensi Pengguna Akhir	1	3	3
4	Pemrosesan Internal yang Kompleks	1	3	3
5	Kemampuan Melakukan Penggunaan Kembali Kode	1	3	3
6	Kemudahan Instalasi	0.5	3	1.5
7	Kemudahaan Penggunaan	0.5	3	1.5
8	Dukungan Antar *Platform*	2	3	6
9	Kemudahan untuk Mengubah	1	3	3
10	Konkurensi	1	3	3
11	Fitur Keamanan Khusus	1	3	3
12	Ketergantungan pada Kode Pihak Ketiga	1	3	3
13	Perlunya pelatihan khusus	1	3	3
Nilai *Technical factor*				42
Nilai Technical Complexity Factor (TCF)		0,6 **+ (**0,01 *** 42) = 1,02**		

Tabel 5.17 menjelaskan mengenai proses penghitungan *Technical Complexity Factor* Sistem DBA *ticketing* yang setiap faktornya bernilai 3. Dari hasil penghitungan pada tabel 5.17 bobot dari masing-masing faktor dikalikan dengan

nilai dari setiap faktor kemudian hasil dari perkalian tersebut dijumlahkan untuk memperoleh nilai *technical factor*. Untuk memeperoleh nilai *Technical Complexity Factor* (TCF) maka menggunakan persamaan (2.6) sehingga hasil *Technical Complexity Factor* (TCF) pada sistem DBA *ticketing* adalah **1,02**.

5.1.3 Menghitung Environmental Factor

Terdapat Terdapat 8 *Environmental factor* pada lembar penilaian yang mana setiap faktor memiliki skala nilai yang telah ditentukan. Skala nilai ditentukan antara 0-5. Penilaian pada *Environmental factor* dilakukan oleh manajer proyek. Lembar penilaian sistem DBA *ticketing* terlampir pada Lampiran. Pada table 5.18 berikut merupakan hasil beserta deskripsi dari nilai yang diberikan pada *environmental factor* sistem DBA *ticketing*.

Tabel 5.18 *Environmental Factor* sistem DBA *ticketing*

No	*Environmental factor*	Nilai	Deskripsi
1	Mengenal metode yang digunakan dalam membuat proyek	2	Satu atau lebih anggota tim pernah menggunakan metode tersebut sekali atau beberapa kali.
2	Pengalaman Aplikasi	2	Beberapa anggota tim mempunyai pengalaman 1 sampai 1,5 tahun, dan anggota tim lainnya adalah pemula.
3	Pengalaman menggunakan pemrograman berorientasi objek	3	Semua anggota tim memiliki 1 sampai 1,5 tahun pengalaman.
4	Menguasai Kemampuan Analisis	2	Memiliki pengalaman dari sedikit proyek.
5	Motivasi	5	Seluruh anggota tim memiliki motivasi yang kuat.
6	Stabilitas kebutuhan proyek	2	*User* meminta berbagai perubahan dilakukan pada berbagai selang waktu.
7	Pekerja paruh waktu	0	Tidak ada pekerja paruh waktu dalam tim.
8	Kesulitan dalam Bahasa Pemrogaman	2	Semua anggota tim memiliki lebih dari 1,5 tahun pengalaman.

Pada table 5.18 merupakan hasil penilaian yang diberikan oleh manajer proyek terhadap *environmental factor* yang kemudian akan digunakan untuk mendapatkan nilai *Environmental Complexity Factor* (ECF). Untuk mendapatkan nilai *Environmental Complexity Factor* (ECF) dengan cara mengkalikan nilai skala yang diberikan oleh manajer proyek dengan bobot setiap faktor pada *environmental factor*.

Tabel 5.19 Penghitungan *Environmental Complexity Factor* (ECF)

Environmental Complexity Factor		**Bobot**	**Nilai**	**Bobot* Nilai**
1	Mengenal metode pengembangan yang digunakan	1,5	2	3
2	Pengalaman Aplikasi	0,5	2	1
3	Pengalaman menggunakan pemrograman berorientasi objek	1	3	3
4	Menguasai Kemampuan Analisis	0,5	2	1
5	Motivasi	1	5	5
6	Stabilitas kebutuhan proyek	2	2	4
7	Pekerja paruh waktu	-1	0	0
8	Kesulitan dalam Bahasa Pemrogaman	-1	2	-2
Total *Environmental factor*				**15**
nilai *Environmental Complexity Factor* (ECF)		**1.4 + (-0,03 * 15) = 0,95**		

Tabel 5.19 merupakan hasil penghitungan dari nilai *Environmental Complexity Factor* (ECF). Pada tabel 5.19 nilai bobot dari masing-masing faktor dikalikan dengan nilai dari setiap faktor yang didapatkan dari lembar penilaian yang diisi oleh manajer proyek kemudian hasil dari perkalian tersebut dijumlahkan untuk memperoleh nilai *environmental factor*. Untuk memeperoleh nilai *Environmental Complexity Factor* (ECF) maka menggunakan persamaan (2.8) sehingga hasil *Environmental Complexity Factor* (ECF) pada sistem DBA *ticketing* adalah **0,95**.

5.1.4 Menghitung Use Case point Sistem DBA Ticketing

Tabel 5.20 Perhitungan *Use Case Point* DBAticketing

Penghitungan	**Hasil**
Unadjusted Use Case Point (UUCP)	157
Technical Complexity Factor (TCF)	1,02
Environmental Complexity Factor (ECF)	0,95
***Use Case point* (UCP)**	157 * 1,02 * 0,95 = **152,133**

Pada Tabel 5.20 di atas, dijelaskan bahwa nilai dari *Unadjusted Use Case Point* (UUCP) sistem DBA ticketing adalah **157**, nilai dari TCF adalah **1,02**, dan nilai dari ECF adalah **0,95**. Nilai *use case point* (UCP) didapatkan dengan mengkalikan nilai dari *Unadjusted Use Case Point* (UUCP)**,** *Technical Complexity Factor* (TCF), dan *Environmental Complexity Factor* (ECF) sehingga didapatkan nilai *Use Case*

Point (UCP) sebersar **152,133**. Dalam metode ANN *model* based on use case point nilai UCP digunakan sebagai salah satu faktor untuk menghitung *effort*.

5.2 Menghitung *Unadjusted Use Case Point* Sistem Pintu Air

Untuk mendapatkan nilai *unadjusted use case point* didapatkan dari penjumlahan nilai *unadjusted use case weight* (UUCW) dengan nilai *unadjusted actor weight* (UAW) mengacu pada persamaan 2. Menghitung *unadjusted use case weight* dibutuhkan jumlah transaksi di setiap *use case*. Untuk menghitung *unadjusted actor weight* diperlukan informasi mengenai antarmuka yang digunakan aktor saat berinteraksi dengan sistem. Aktor akan dikategorikan dengan melihat bagaimana cara aktor berinteraksi dengan sistem sesuai pada table 5.21. Berikut adalah tabel penghitungan nilai *Unadjusted Actor Weight* sistem pintu air.

Tabel 5.21 *Unadjusted Actor Weight Sistem Pintu Air*

No	Nama Aktor	Kategori Aktor	Bobot Aktor
1	Admin	Kompleks	3
Total UAW			3

Menurut tabel 5.21 aktor Admin masuk kedalam kategori kompleks dan memiliki nilai bobot aktor sebesar 3. Hal tersebut dikarenakan aktor Admin berinteraksi dengan sistem melalui *Graphical User Interface* (GUI). Total nilai *Unadjusted Actor Weight* (UAW) pada sistem pintu air adalah **3**.

5.2.1 Transaksi Use Case

Transaksi dapat dihitung dengan cara menghitung jumlah langkah yang ada pada *basic flow* dan *alternative flow* yang ada didalam sebuah use case (Kristen Ribu, 2001). Terdapat tiga kategori yang berdasarkan jumlah dari transaksi yang terjadi dalam *use case.* Kategori tersebut yaitu sederhana, medium atau rata-rata, atau kompleks. Berikut merupakan transaksi yang ada pada sistem pintu air.

1. Transaksi *use case login*

Tabel 5.22 Transaksi *use case Login*

Basic flow : 1. *Use case* dimulai ketika aktor pengguna mengisi *form login* berupa nama dan sandi. 2. Aktor pengguna memasukkan nama dan sandi ke dalam sistem. 3. Sistem melakukan identifikasi terhadap pengguna. 4. Aktor masuk kedalam sistem. *Alternative Flow* : 1. Menangani adanya kesalahan dalam proses identifikasi pengguna

Jumlah Transaksi: 5

Berdasarkan tabel 5.22 jumlah transaksi yang terjadi pada *use case login* adalah 5 transaksi, hal tersebut dihitung berdasarkan 4 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case login*.

2. Transaksi *use case* melihat data ketinggian air

Tabel 5.23 Transaksi Use Case Melihat data ketinggian air

Basic flow: 1. *Use case* mulai saat aktor memilih untuk melihat data ketinggian air. 2. Sistem memuat data ketinggian air. 3. Sistem menampilkan data ketinggian air. 4. Aktor memilih untuk mencari data sesuai kata kunci pada data ketinggian air. 5. Aktor memasukkan kata kunci sesuai yang ingin dicari. 6. Sistem memuat data yang dicari. 7. Sistem menampilkan data yang dicari pada laporan. *Alternative Flow* : 1. Menangani kondisi ketika data yang dicari tidak ditemukan
Jumlah transaksi : 8

Berdasarkan tabel 5.23 jumlah transaksi yang terjadi pada *use case* melihat ketinggiar air adalah 8 transaksi, hal tersebut dihitung berdasarkan 7 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* melihat ketinggan air.

3. Transaksi *use case* menambah data ketinggian air

Tabel 5.24 Transaksi Use Case menambah data ketinggian air

Basic flow: 1. *Use case* mulai saat aktor memilih melihat data ketinggian air. 2. Sistem memuat data ketinggian air. 3. Sistem menampilkan data ketinggian air. 4. Aktor memilih untuk menambah data ketinggian air. 5. Sistem menampilkan *form* penambahan data ketinggian air baru. 6. Aktor mengisi *form* penambahan data ketinggian air baru. 7. Aktor memilih menyimpan data ketinggian air baru. 8. Sistem memproses penyimpanan data ketinggian air baru. 9. Sistem menampilkan pesan bahwa data ketinggian air baru berhasil disimpan. *Alternative Flow* : 1. Menangani kegagalan penambahan data ketinggian air baru.

Jumlah transaksi : 10

Berdasarkan tabel 5.24 jumlah transaksi yang terjadi pada *use case* Menambah data ketinggain air adalah 10 transaksi, hal tersebut dihitung berdasarkan 9 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* Menambah data ketinggain air.

4. Transaksi *use case* mengubah data ketinggian air

Tabel 5.25 Transaksi Use Case Mengubah Data ketinggian air

Basic flow: 1. *Use case* dimulai ketika aktor memilih untuk melihat data ketinggian air 2. Sistem memuat data memuat data ketinggian air. 3. Sistem menampilkan data ketinggian air. 4. Aktor memilih data ketinggian air yang akan diubah. 5. Sistem menampilkan *form* untuk mengubah data. 6. Aktor mengubah data. 7. Aktor memilih memperbarui data ketinggian air. 8. Sistem melakukan proses pengubahan data ketinggian air. 9. Sistem mengeluarkan pesan bahwa data telah berhasil diperbarui. *Alternative Flow* : a. Menangani kondisi ketika data ketinggian air gagal diperbarui.
Jumlah transaksi : 10

Berdasarkan tabel 5.25 jumlah transaksi yang terjadi pada *use case* Mengubah data ketinggain air adalah 10 transaksi, hal tersebut dihitung berdasarkan 9 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* Mengubah data ketinggain air.

5. Transaksi *use case* menghapus data ketinggian air

Tabel 5.26 Transaksi Use Case Menghapus Data Ketinggian Air

Basic flow: 1. *Use case* dimulai ketika aktor memilih untuk melihat data ketinggian air. 2. Sistem memuat data ketinggian air. 3. Sistem menampilkan data ketinggian air. 4. Aktor memilih data yang akan dihapus. 5. Sistem melakukan proses penghapusan data ketinggian air. 6. Sistem menampilkan pesan bahwa data ketinggian air telah berhasil dihapus. *Alternative Flow* : 1. Menangani kondisi ketika data ketinggian air gagal dihapus.
Jumlah transaksi : 7

Berdasarkan tabel 5.26 jumlah transaksi yang terjadi pada *use case* Menghapus data ketinggain air adalah 7 transaksi, hal tersebut dihitung berdasarkan 6 langkah yang ada pada *basic flow* dan 1 langkah yang ada pada *alternative flow* pada *use case* Menghapus data ketinggain air.

Tabel 5.27 Perhitungan UUCW Sistem Pintu Air

Jumlah transaksi pada *use case*	Komplesitas	Jumlah *use case*	Bobot	Bobot * Jumlah *Use case*
Kurang dari 4 transaksi	Sederhana	0	5	0
4 hingga 7 transaksi	Medium	2	7	14
Lebih dari 7 transaksi	Kompleks	3	15	45
Total UUCW				59

Menurut tabel 5.27 pada sistem Pintu air memiliki 2 *use case* yang memiliki jumlah transaksi antara 4 hingga 7 transaksi sehingga 2 *use case* tersebut masuk kedalam kategori medium, kemudian 3 *use case* memiliki jumlah transaksi lebih dari 7 sehingga masuk kedalam kategori kompleks. Nilai *Unadjusted Use Case Weight* (UUCW) pada sistem Pintu Air didapat dengan mengalikan bobot setiap tingkat transaksi dengan jumlah *use case* pada setiap tingkat transaksi dan hasil nilai *Unadjusted Use Case Weight* (UUCW) adalah **59**.

Tabel 5.28 Penghitungan *Unadjusted Use Case Point* Sistem Pintu Air

Penghitungan	Hasil
Unadjusted Use Case Weight (UUCW)	59
Unadjusted Actor Weight (UAW)	3
UUCW + UAW =	62

Dengan hasil penghitungan *unadjusted use case weight* (UUCW) dan *unadjusted actor weight* (UAW), dapat diperoleh nilai *unadjusted use case point* pada sistem Pintu Air. Pada Tabel 5.15 menjumlahkan nilai *unadjusted use case weight* dan *unadjusted actor weight* sehingga ditemukan bahwa nilai *Unadjusted Use Case Point* (UUCP) untuk sistem Pintu Air adalah **62.**

5.2.2 Menghitung Technical Complexity Factor

Pada penelitian ini penulis memberikan nilai 3 terhadap *technical factor* karena sesuai dengan penelitian *(*Ani & Basri, 2013) pada penelitian tersebut menjelaskan alasan utama mengapa nilai 3 diberikan pada *technical factor* adalah karena *use case* yang ada pada sistem kurang dari 50 *use case*. Hal tersebut berarti sistem yang dikembangkan tidak rumit. Jumlah *use case* pada sistem Pintu Air adalah 12 *use case,* maka penulis menetapkan nilai 3 pada *technical factor*. Nilai 3 pada *technical factor* berarti faktor tersebut memiliki pengaruh yang biasa saja terhadap pengembangan sistem.

Tabel 5.29 Penilaian *technical factor* Pintu Air

Technical factor		**Nilai**	**Deskripsi**
1	Sistem Terdistribusi	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
2	Performa (respon time)	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
3	Efisiensi dari *user interface*	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
4	Kompleksitas proses	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
5	Penggunaan kembali kode program	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
6	Kemudahan Instalasi	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
7	Kemudahaan Pengoperasian	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
8	Dapat diaplikasikan diberbagai platform	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
9	Mudah untuk dirubah	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
10	Konkurensi	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
11	Fitur Keamanan	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
12	Ketergantungan pada Pihak Ketiga	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.
13	Perlunya Pelatihan Khusus	3	*Technical factor* berpengaruh biasa saja terhadap pengembangan proyek.

Pada tabel 5.29 berisi tentang deskripsi penilaian terhadap 13 faktor pada *technical factor* sistem pintu air. Tabel 5.29 menjelaskan mengenai penghitungan *Technical Complexity Factor* Sistem Pintu air yang diperoleh dari hasil perkalian bobot setiap *technical factor* dengan nilai *technical factor.*

Tabel 5.30 Penghitungan technical Factor

Technical factor		**Bobot**	**Nilai**	**Bobot * Nilai**
1	Sistem Terdistribusi	2	3	6
2	Kinerja	1	3	3

Technical factor		Bobot	Nilai	Bobot * Nilai
3	Efisiensi Pengguna Akhir	1	3	3
4	Pemrosesan Internal yang Kompleks	1	3	3
5	Kemampuan Melakukan Penggunaan Kembali Kode	1	3	3
6	Kemudahan Instalasi	0.5	3	1.5
7	Kemudahaan Penggunaan	0.5	3	1.5
8	Dukungan Antar *Platform*	2	3	6
9	Kemudahan untuk Mengubah	1	3	3
10	Konkurensi	1	3	3
11	Fitur Keamanan Khusus	1	3	3
12	Ketergantungan pada Kode Pihak Ketiga	1	3	3
13	Fasilitas Pelatihan Khusus untuk Pengguna Diperlukan	1	3	3
Nilai *Technical factor*				**42**
Technical Complexity Factor (TCF)		0,6 **+ (** 0,01 *** 42) = 1,02**		

Dari hasil perhitungan pada tabel 5.30 bobot dari masing-masing faktor dikalikan dengan nilai dari setiap faktor kemudian hasil dari perkalian tersebut dijumlahkan untuk memperoleh nilai *technical factor*. Untuk memeperoleh nilai *Technical Complexity Factor* (TCF) maka menggunakan persamaan (2.6) sehingga hasil *Technical Complexity Factor* (TCF) pada sistem Pintu air adalah **1,02**.

5.2.3 Menghitung Environmental Factor

Terdapat 8 *Environmental factor* pada lembar penilaian yang mana setiap faktor memiliki skala nilai yang telah ditentukan. Skala nilai ditentukan antara 0 - 5. Penilaian pada *Environmental factor* dilakukan oleh manajer proyek. Lembar penilaian sistem Pintu air terlampir pada Lampiran.

Tabel 5.31 *Environmental Factor* sistem Pintu Air

No	*Environmental factor*	Nilai	Deskripsi
1	Mengenal metode yang digunakan dalam membuat proyek	2	Satu atau lebih anggota tim pernah menggunakan metode tersebut sekali atau beberapa kali.
2	Pengalaman Aplikasi	3	Semua anggota tim mempunyai pengalaman lebih dari 1,5 tahun.

3	Pengalaman menggunakan pemrograman berorientasi objek	3	Semua anggota tim mempunyai 1 sampai 1,5 tahun pengalaman.
4	Menguasai Kemampuan Analisis	2	Memiliki pengalaman dari sedikit proyek.
5	Motivasi	5	Seluruh anggota tim memiliki motivasi yang kuat.
6	Stabilitas kebutuhan proyek	3	Kestabilan masih membutuhkan perubahan kecil.
7	Pekerja paruh waktu	0	Tidak ada pekerja paruh waktu dalam tim.
8	Kesulitan dalam Bahasa Pemrogaman	2	Semua anggota tim mempunyai lebih dari 1,5 tahun pengalaman.

Pada tabel 5.31 berisi megenai nilai beserta deskripsi penilaian terhadap *environmental factor* sistem Pintu air. Nilai pada tabel 5.31 adalah hasil penilaian oleh manajer proyek terhadap *environmental factor* yang hasil penilaian tersebut akan diolah untuk mendapatkan nilai *Environmental Complexity Factor* (ECF). Untuk mendapatkan nilai *Environmental Complexity Factor* (ECF) dengan cara mengalikan nilai skala yang diberikan oleh manajer proyek dengan bobot setiap *environmental factor* pada sistem Pintu air.

Tabel 5.32 Penghitungan *Environmental Complexity Factor* (ECF)

Environmental Complexity Factor		**Bobot**	**Nilai**	**Bobot* Nilai**
1	Mengenal metode pengembangan yang digunakan	1,5	2	3
2	Pengalaman Aplikasi	0,5	3	1,5
3	Pengalaman menggunakan pemrograman berorientasi objek	1	3	3
4	Menguasai Kemampuan Analisis	0,5	2	1
5	Motivasi	1	5	5
6	Stabilitas kebutuhan proyek	2	3	6
7	Pekerja paruh waktu	-1	0	0
8	Kesulitan dalam Bahasa Pemrogaman	-1	2	-2
Nilai *Environmental factor*				**17,5**
Nilai *Environmental Complexity Factor* (ECF)		1.4 **+ (** -0,03 *** 17,5) = 0,875**		

Pada tabel 5.32 nilai bobot dari masing-masing faktor dikalikan dengan nilai dari setiap faktor yang didapatkan dari lembar penilaian yang diisi oleh manajer proyek sistem Pintu air kemudian hasil dari perkalian tersebut dijumlahkan untuk

memperoleh nilai *environmental factor*. Untuk memeperoleh nilai *Environmental Complexity Factor* (ECF) maka menggunakan persamaan (2.8) sehingga hasil *Environmental Complexity Factor* (ECF) pada sistem Pintu air adalah **0,875**.

1.1.2 Menghitung Use Case point

Tabel 5.33 Penghitungan *Use Case Point* Pintu Air

Perhitungan	**Hasil**
Unadjusted Use Case Point (UUCP)	62
Technical Complexity Factor (TCF)	1,02
Environmental Complexity Factor (ECF)	0,875
***Use Case point* (UCP)**	62 * 1,02 * 0,875 = 55,335

Pada Tabel 5.20 di atas, dijelaskan bahwa nilai dari *unadjusted use case point* (UUCP) sistem Pintu air adalah **62**, nilai dari *technical complexity factor* adalah **1,02**, dan nilai dari *environmental complexity factor* adalah **0,875**. Nilai *use case point* didapatkan dengan mengkalikan nilai dari *unadjusted use uase point*, *technical complexity factor* dan *environmental complexity factor* sehingga didapatkan nilai *Use Case Point* (UCP) sebersar **55,335**. Dalam metode *ANN model based on use case point* nilai UCP digunakan sebagai salah satu faktor untuk menghitung *effort*.

5.3 Menghitung Nilai Produktifitas

Nilai produktifitas pada metode *ANN model based on Use Case Point* menggunakan nilai dari *Environmental complexity factor* (ECF) (B. Nassif, 2011). Sebelumnya pada penelitian ini nilai *Environmental complexity factor* (ECF) sudah didapatkan karena nilai *Environmental complexity factor* (ECF) merupakan salah satu komponen yang digunakan untuk meghitung nilai *Use Case Point* (UCP).

5.3.1 Penilaian Produktifitas sistem DBA *ticketing*

Berikut merupakan hasil perhitungan penilaian produktifitas sistem DBA *ticketing*.

Tabel 5.34 Perhitungan *Environmental Complexity Factor* (ECF)

Environmental Complexity Factor		**Bobot**	**Nilai**	**Bobot* Nilai**
1	Mengenal metode yang digunakan dalam membuat proyek	1,5	2	3
2	Pengalaman Aplikasi	0,5	2	1
3	Pengalaman menggunakan pemrograman berorientasi objek	1	3	3
4	Menguasai Kemampuan Analisis	0,5	2	1
5	Motivasi	1	5	5

6	Stabilitas kebutuhan proyek	2	2	4
7	Pekerja paruh waktu	-1	0	0
8	Kesulitan dalam Bahasa Pemrogaman	-1	2	-2
Total *Environmental factor*				**15**
Nilai *Environmental Complexity Factor* (ECF)		**1.4 + (-0,03 * 16,5) = 0,95**		

Pada tabel 5.34 nilai bobot dari masing-masing faktor dikalikan dengan nilai dari setiap faktor yang didapatkan dari lembar penilaian yang diisi oleh manajer proyek kemudian hasil dari perkalian tersebut dijumlahkan untuk memperoleh nilai *environmental factor*. Untuk memeperoleh nilai *Environmental Complexity Factor* (ECF) maka menggunakan persamaan (2.8) sehingga hasil *Environmental Complexity Factor* (ECF) pada sistem DBA *ticketing* adalah 0,95, sehingga nilai produktifitas dari sistem DBA ticketing adalah **0,95**

5.3.2 Penilaian Produktifitas Sistem Pintu Air

Berikut merupakan hasil perhitungan penilaian produktifitas sistem Pintu Air.

Tabel 5.35 Perhitungan *Environmental Complexity Factor* (ECF)

Environmental Complexity Factor		**Bobot**	**Nilai**	**Bobot* Nilai**
1	Mengenal metode yang digunakan dalam membuat proyek	1,5	2	3
2	Pengalaman Aplikasi	0,5	3	1,5
3	Pengalaman menggunakan pemrograman berorientasi objek	1	3	3
4	Menguasai Kemampuan Analisis	0,5	2	1
5	Motivasi	1	5	5
6	Stabilitas kebutuhan proyek	2	3	6
7	Pekerja paruh waktu	-1	0	0
8	Kesulitan dalam Bahasa Pemrogaman	-1	2	-2
N*ilai Environmental factor*				**17,5**
Nilai *Environmental Complexity Factor* (ECF)		**1.4 + (-0,03 * 17,5) = 0,875**		

Pada tabel 5.35 nilai bobot dari masing-masing faktor dikalikan dengan nilai dari setiap faktor yang didapatkan dari lembar penilaian yang diisi oleh manajer proyek kemudian hasil dari perkalian tersebut dijumlahkan untuk memperoleh nilai *environmental factor*. Untuk memeperoleh nilai *Environmental Complexity Factor* (ECF) maka menggunakan persamaan (2.8) sehingga hasil *Environmental Complexity Factor* (ECF) pada sistem Pintu air adalah 0,875, sehingga nilai produktifitas dari sistem Pintu air adalah **0,875**.

5.4 Menghitung Bobot Kompleksitas

Nilai kompleksitas diukur melalui 5 level pada metode *ANN model based on use case point* (A. B. Nassif, 2012) dengan masing – masing level memiliki karakteristik tersendiri. Berdasarkan hasil wawancara dengan pihak CV. Profile Image Studio diketahui bahwa sistem DBA ticketing masuk kedalam level 2, karena diketahui bahwa sekitar 10% dari desain atau bagian proyek sistem berasal dari proyek yang serupa sehingga, bobot kompleksitas sistem DBA ticketing adalah 2. Sedangkan untuk sistem Pintu air masuk kedalam level 3, karena diketahui bahwa tidak ada bagian dari proyek yang mengimplementasi proyek sebelumnya sehingga, bobot kompleksitas sistem pintu air adalah 3, hal tersebut ditulis dalam tabel 5.36 berikut.

Tabel 5.36 Bobot Kompleksitas Sistem

Sistem	Level	Bobot Kompleksitas
DBA ticketing	2	2
Pintu Air	3	3

5.5 Menghitung Effort berdasarkan metode *ANN model based on Use Case Point*

Nilai effort berdasarkan *ANN model based on use case point* dihitung dengan memasukkan nilai *Use Case Point* (UCP), nilai produktifitas dan nilai bobot kompleksitas kedalam persamaan 2.2, dan berikut merupakan perhitungan effort sistem DBA *ticketing* menggunakan *ANN model based on use case point* :

Tabel 5.37 *Perhitungan Effort* sistem DBA ticketing

Perhitungan	Hasil
Use Case Point (UCP)	**152,133**
Produktifitas	**0,95**
Kompleksitas	**2**
***Nilai effort atau* usaha**	3.53 + (0.88 x **152,133**) – (0.009 x **0,95**) + (0.31 x **2**) = **138,01**

Pada tabel 5.37 dijelaskan bahwa nilai dari UCP sebesar **152,133**, kemudian nilai Produktifitas adalah **0,95** dan nilai kompleksitas adalah **2**. Berdasarkan hasil perhitungan menggunakan metode *ANN model based on use case point* didapatkan bahwa hasil effort sistem DBA *ticketing* yang didapat adalah **138,01** dalam satuan person-hours.

Tabel 5.38 *Perhitungan Effort* sistem Pintu Air

Perhitungan	Hasil
Use Case Point (UCP)	**55,335**

Produktifitas	**0,95**
Kompleksitas	**3**
***Nilai effort atau* usaha**	3.53 + (0.88 x **55,335**) – (0.009 x **0,875**) + (0.31 x **3**) = **53,14**

Pada tabel 5.38 dijelaskan bahwa nilai dari UCP sebesar **55,335**, kemudian nilai Produktifitas adalah **0,875** dan nilai kompleksitas adalah **3**. Berdasarkan hasil perhitungan menggunakan metode *ANN model based on use case point* didapatkan bahwa hasil effort sistem Pintu air yang didapat adalah **53,14** dalam satuan person-hours.

5.6 Sistem DBA ticketing

5.6.1 Distribusi Effort

Nilai effort yang sudah didapatkan kemudian didistribusikan pada masing-masing aktivitas dengan mengalikan persentase effort setiap aktivitas (Saleh, 2011) dengan total nilai effort yang telah dihitung sebelumnya. Hasilnya berupa nilai effort pada masing-masing aktivitas pengembangan perangkat lunak yang kemudian akan digunakan untuk menghitung jumlah sumber daya manusia (staff), durasi waktu dan jumlah biaya. Total nilai effort dari sistem DBA ticketing adalah 138,01 staff hours. Nilai tersebut akan didistribusikan kepada setiap aktivitas pengembangan perangkat lunak seperti yang ada pada tabel 5.39 dibawah ini.

Tabel 5.39 Distribusi Effort sistem DBA ticketing

Aktivitas	% Effort	Effort
Software Phases		
Requirements	7,5	10.35
Spesifications	7,5	10.35
Design	10	13.8
Implementation	10	13.8
Integration Testing	7,5	10.35
Acceptance & Deployment	7,5	10.35
Ongoing life-cycle activities		
Project Management	8,34	11.5
Configuration Management	4,16	5.7
Quality Assurance	8,34	11.5
Documentation	4,16	5.7
Training and support	4,16	5.7
Evaluation and Testing	20,84	28.7
Total Effort		**138,01 (*staff hours*)**

Tabel 5.39 merupakan distribusi effort pada masing - masing aktivitas yang ada sesuai dengan persentase effort setiap aktivitas yang ditentukan pada *guideline* distribusi effort hasil penelitian Kassem Saleh (2011) dan ketika nilai effort pada

masing - masing aktivitas dijumlah maka totalnya akan sama dengan 138,01 staff hours.

5.6.2 Menghitung Durasi waktu

Durasi waktu pengembangan sistem DBA ticketing didapatkan dengan memasukkan nilai UCP dari sistem sistem DBA ticketing yaitu 152,13 kedalam persamaan 2.1.

Tabel 5.40 Perhitungan Durasi waktu DBA ticketing

UCP	152,13
p	8,2
h	8
Durasi waktu	((8,2 * 152,13) / 8) = **155,9** hari

Pada tabel 5.40 dijelaskan bahwa nilai dari UCP sebesar 152,133, kemudian dikalikan dengan 8,2 dan dibagi dengan 8. Berdasarkan hasil perhitungan tersebut didapatkan bahwa durasi waktu adalah 155,9 dalam satuan hari. Kemudian durasi waktu dalam satuan hari tersebut dirubah menjadi satuan jam , dalam penelitian ini jumlah jam kerja dalam satu hari adalah 8 jam maka dengan mengalikan 155,9 dengan 8 akan didapatkan hasil yaitu **1247,46** jam.

Tabel 5.41 Jumlah sumber daya manusia sistem DBA ticketing

Aktivitas	**% Effort**	**hours**	**Staff**
Software Phases			
Requirements	7,5	93,5	1
Spesifications	7,5	93,5	1
Design	10	124,7	1
Implementation	10	124,7	1
Integration Testing	7,5	93,5	1
Acceptance & Deployment	7,5	93,5	1
Ongoing life-cycle activities			
Project Management	8,34	104	1
Configuration Management	4,16	51,8	1
Quality Assurance	8,34	104	1
Documentation	4,16	51,8	1
Training and support	4,16	51,8	1
Evaluation and Testing	20,84	259,9	1
Total		**1247,46**	**12**

Pada tabel 5.41 total durasi waktu yaitu 1247,46 jam dikalikan dengan masing-masing %Effort pada setiap aktivitas maka akan diketahui durasi waktu yang dibutuhkan pada masing-masing aktivitas pengembangan sistem DBA ticketing. Jumlah staff pada masing-maisng aktivitas bersal dari nilai effort masing-masing aktivitas pengembangan sistem DBA ticketing (Tabel 5.39) dibagi dengan durasi

waktu pada masing-masing aktivitas sehingga total sumber daya manusia (*staff*) yang dibutuhkan berjumlah **12** orang.

5.6.3 Menghitung Total Biaya

Tabel 5.42 UMK Kota Malang 2017

UMK Kota Malang 2017	Standar Gaji Per Bulan (Rp)	Standar Gaji Per Jam (Rp)
	Rp 2.272.167	Rp 14.200

Pada Tabel 5.42 di atas, standar gaji mengacu pada UMK kota Malang pada tahun 2017. Standar gaji per jam didapatkan dengan membagi standar gaji per bulan dengan jumlah jam kerja perusahaan selama 1 bulan. Di dalam 1 bulan jumlah hari kerja pada CV. Profile Image Studio adalah 20 hari kerja dan jumlah jam kerja setiap harinya adalah 8 jam kerja. Jumlah jam kerja selam 1 bulan adalah 160 jam kerja yang didapat dari 20 dikalikan dengan 8. Jika standar gaji per bulan dibagi dengan 160, maka akan didapatkan hasilnya yaitu standar gaji Per Jam yaitu sebesar Rp 14.200. Menghitung total biaya berarti menjumlahkan semua biaya per aktivitas pada *software phases* dan *ongoing life cycle.* Setelah mendapatkan durasi waktu setiap aktivitas dan jumlah sumber daya (staff) kemudian dapat dihitung biaya per aktivitasnya. Pada tabel 5.43 berikut menampilkan total biaya per aktivitas pada pengembangan sistem pintu air.

Tabel 5.43 Biaya sistem DBA ticketing

Aktivitas	Jumlah staff	Durasi waktu (jam)	Standar gaji per jam (Rp)	Biaya per staff (Rp)	Biaya per Aktivitas (Rp)
Software Phases					
Requirements	1	93,5	14.200	1.327.700	1.327.700
Spesifications	1	93,5	14.200	1.327.700	1.327.700
Design	1	124,7	14.200	1.770.740	1.770.740
Implementation	1	124,7	14.200	1.770.740	1.770.740
Integration Testing	1	93,5	14.200	1.327.700	1.327.700
Acceptance & Deployment	1	93,5	14.200	1.327.700	1.327.700
Ongoing life-cycle activities					
Project Management	1	104	14.200	1.476.800	1.476.800

Configuration Management	1	51,8	14.200	735.560	735.560
Quality Assurance	1	104	14.200	1.476.800	1.476.800
Documentation	1	51,8	14.200	735.560	735.560
Training and support	1	51,8	14.200	735.560	735.560
Evaluation and Testing	1	259,9	14.200	3.690.580	3.690.580

Pada tabel 5.43 berikut menampilkan total biaya per aktivitas pada pengembangan sistem DBA ticketing dan dapat dilihat bahwa biaya per staff (rupiah) diperoleh dengan mengalikan lamanya durasi waktu (jam) dengan standar gaji per jam (rupiah). Biaya per aktivitas didapatkan dengan mengalikan biaya per staff (rupiah) dengan jumlah staff yang ada pada setiap aktivitas. Nilai estimasi biaya per aktivitas akan digunakan untuk mendapatkan total biaya seluruh aktivitas dalam pengembangan sistem DBA ticketing.

Tabel 5.44 Total Biaya sistem DBA ticketing

Fase	Total Estimasi Biaya
Software Phases	Rp 8.852.280
Ongoing life-cycle activities	Rp 8.850.860
Total	**Rp 17.703.140**

Tabel 5.44 menjelaskan mengenai total biaya yang diperlukan oleh sistem DBA ticketing, yang mana hasil penjumlahan dari aktivitas *Software Phases* adalah sebesar Rp 8.852.280 dan *Ongoing life-cycle activities* sebesar Rp 8.850.860 dan total biaya untuk membuat Sistem Pintu air dengan menggunakan metode *ANN model based on Use Case Point* adalah sebesar **Rp 17.703.140**.

5.7 Sistem Pintu Air

5.7.1 Distribusi Effort

Nilai effort yang sudah didapatkan kemudian didistribusikan pada masing-masing aktivitas dengan mengalikan persentase effort setiap aktivitas (Saleh, 2011) dengan total nilai effort yang telah dihitung sebelumnya. Hasilnya berupa nilai effort pada masing-masing aktivitas pengembangan perangkat lunak yang kemudian akan digunakan untuk menghitung jumlah sumber daya manusia (staff), durasi waktu dan jumlah biaya. Total nilai effort dari sistem Pintu air adalah 53,14 staff hours. Nilai tersebut akan didistribusikan kepada setiap aktivitas pengembangan perangkat lunak seperti yang ada pada tabel 5.45 dibawah ini.

Tabel 5.45 Distribusi Effort sistem Pintu Air

Aktivitas	% Effort	Effort
Software Phases		
Requirements	7,5	3,9
Spesifications	7,5	3,9
Design	10	5,3
Implementation	10	5,3
Integration Testing	7,5	3,9
Acceptance & Deployment	7,5	3,9
Ongoing life-cycle activities		
Project Management	8,34	4,4
Configuration Management	4,16	2,2
Quality Assurance	8,34	4,4
Documentation	4,16	2,2
Training and support	4,16	2,2
Evaluation and Testing	20,84	11
Total Effort		**53,14 (*staff hours*)**

Tabel 5.45 merupakan distribusi effort pada masing - masing aktivitas yang ada sesuai dengan persentase effort setiap aktivitas yang ditentukan pada *guideline* distribusi effort hasil penelitian Kassem Saleh (2011) dan ketika nilai effort pada masing - masing aktivitas dijumlah maka totalnya akan sama dengan 53,14 staff hours.

5.7.2 Menghitung Durasi Waktu

Durasi waktu pengembangan sistem pintu air didapatkan dengan memasukkan nilai UCP dari sistem sistem pintu air yaitu 55,335 kedalam persamaan 2.1

Tabel 5.46 Perhitungan durasi waktu sistem Pintu Air

UCP	55,33
p	8,2
h	8
Durasi waktu	((8,2 * 55,33) / 8) = **56,7** hari

Pada tabel 5.46 dijelaskan bahwa nilai dari UCP sebesar 55,33, kemudian dikalikan dengan 8,2 dan dibagi dengan 8. Berdasarkan hasil perhitungan tersebut didapatkan bahwa durasi waktu adalah 56,7 dalam satuan hari. Kemudian durasi waktu dalam satuan hari tersebut dirubah menjadi satuan jam , dalam penelitian ini jumlah jam kerja dalam satu hari adalah 8 jam maka dengan mengalikan 56,7 dengan 8 akan didapatkan hasil yaitu **453,6** jam.

Tabel 5.47 Jumlah sumber daya manusia sistem Pintu Air

Aktivitas	% Effort	hours	Staff

Software Phases			
Requirements	7,5	34,02	1
Spesifications	7,5	34,02	1
Design	10	45,36	1
Implementation	10	45,36	1
Integration Testing	7,5	34,02	1
Acceptance & Deployment	7,5	34,02	1
Ongoing life-cycle activities			
Project Management	8,34	37,8	1
Configuration Management	4,16	18,8	1
Quality Assurance	8,34	37,8	1
Documentation	4,16	18,8	1
Training and support	4,16	18,8	1
Evaluation and Testing	20,84	94,5	1
Total		**453,6**	**12**

Pada tabel 5.47 total durasi waktu yaitu 453,6 jam dikalikan dengan masing-masing %Effort pada setiap aktivitas maka akan diketahui durasi waktu yang dibutuhkan pada masing-masing aktivitas pengembangan sistem Pintu Air. Jumlah staff pada masing-maisng aktivitas bersal dari nilai effort masing-masing aktivitas pengembangan sistem pintu air (Tabel 5.40) dibagi dengan durasi waktu pada masing-masing aktivitas sehingga total sumber daya manusia (*staff*) yang dibutuhkan berjumlah **12** orang.

5.7.3 Menghitung Total Biaya

Menghitung total biaya berarti menjumlahkan semua biaya per aktivitas pada *software phases* dan *ongoing life cycle.* Setelah mendapatkan durasi waktu setiap aktivitas dan jumlah sumber daya (staff) kemudian dapat dihitung biaya per aktivitasnya. Pada tabel 5.48 berikut menampilkan total biaya per aktivitas pada pengembangan sistem pintu air.

Tabel 5.48 Biaya sitem Pintu Air

Aktivitas	Jumlah staff	Durasi (jam)	Standar gaji per jam (Rp)	Biaya per staf (Rp)	Biaya per Aktivitas (Rp)
Software Phases					
Requirements	1	34,02	14.200	483.084	483.084
Spesifications	1	34,02	14.200	483.084	483.084
Design	1	45,36	14.200	644.112	644.112
Implementation	1	45,36	14.200	644.112	644.112
Integration Testing	1	34,02	14.200	483.084	483.084

Acceptance & Deployment	1	34,02	14.200	483.084	483.084
Ongoing life-cycle activities					
Project Management	1	37,8	14.200	536.760	536.760
Configuration Management	1	18,8	14.200	266.960	266.960
Quality Assurance	1	37,8	14.200	536.760	536.760
Documentation	1	18,8	14.200	266.960	266.960
Training and support	1	18,8	14.200	266.960	266.960
Evaluation and Testing	1	94,5	14.200	1.341.900	1.341.900

Pada tabel 5.48 berikut menampilkan total biaya per aktivitas pada pengembangan sistem pintu air dan dapat dilihat bahwa biaya per staff (rupiah) diperoleh dengan mengalikan lamanya durasi waktu (jam) dengan standar gaji per jam (rupiah). Biaya per aktivitas didapatkan dengan mengalikan biaya per staff (rupiah) dengan jumlah staff yang ada pada setiap aktivitas. Nilai estimasi biaya per aktivitas digunakan untuk mendapatkan total biaya seluruh aktivitas dalam pengembangan sistem pintu air.

Tabel 5.49 Total biaya sistem Pintu Air

Fase	Total Estimasi Biaya
Software Phases	Rp 3.220.560
Ongoing life-cycle activities	Rp 3.216.300
TOTAL	**Rp 6.436.860**

Pada Tabel 5.49 hasil penjumlahan dari aktivitas *Software Phases* adalah sebesar Rp.220.560 dan *Ongoing life-cycle activities* sebesar Rp 3.216.300 maka total biaya untuk membuat Sistem Pintu air dengan menggunakan metode *ANN model based on Use Case Point* adalah sebesar **Rp 6.436.860**.

BAB 6 PENUTUP

Berdasarkan hasil implementasi metode *ANN model based on use case point* yang sudah dilakukan dalam menghitung biaya pengembangan sistem DBA *ticketing* dan sistem Pintu Air maka didapatkan beberapa kesimpulan dan penulis menambahkan saran bagi penelitian lebih lanjut.

6.1 Kesimpulan

Kesimpulan yang dapat diambil adalah sebagai berikut:

1. Estimasi effort atau usaha yang diperoleh dengan menggunakan metode *ANN model based on use case point* pada Sistem DBA ticketing adalah 138,02 (*staff hours*). Sedangkan estimasi effort atau usaha yang diperoleh dengan menggunakan metode *ANN model based on use case point* pada Sistem Pintu Air adalah 53,14 (*staff hours*).
2. Estimasi jumlah sumber daya manusia (staff) untuk pengembangan sistem DBA ticketing berjumlah 12 orang dan estimasi jumlah sumber daya manusia (staff) untuk pengembangan sistem Pintu Air berjumlah 12 orang.
3. Estimasi durasi waktu pengembangan sistem DBA ticketing adalah 1247,46 jam sedangkan untuk estimasi durasi waktu pengembangan sistem Pintu Air adalah 453,6 jam.
4. Estimasi biaya pengembangan Sistem DBA ticketing dengan mengimplementasikan metode *ANN model based on Use Case Point* adalah sebesar Rp 17.703.140. Sedangkan estimasi biaya pengembangan Sistem Pintu Air dengan mengimplementasikan metode *ANN model based on Use Case Point* adalah sebesar Rp 6.436.860.

6.2 Saran

Dalam penelitian ini penulis memberi nilai 3 pada setiap faktor yang terdapat pada *technical factor*, karena penulis menggunakan hasil penelitian Ani dan Basri (2013). Mereka memberikan nilai 3 pada setiap faktor jika jumlah use case pada sistem kurang dari 50 dan jumlah use case pada sistem di dalam penilitian ini kurang dari 50. Hal tersebut mempengaruhi hasil estimasi sumber daya manusia, waktu serta biaya. Pada penelitian lanjutan, penulis menyarankan dalam menetapkan *technical factor* manajer proyek dapat menetapkan skala dari 0-5 pada setiap faktor yang ada.

DAFTAR PUSTAKA

Anda, B., 2002. Comparing effort estimates based on use cases with expert estimates. *Empirical Assessment in SoftwareEngineering* (EASE), (p. 13). Keele UK.

Anda, B., Dreiem, H., Sjoberg, D., dan Jorgensen, M., 2001. Estimating software development effort based on use cases - experiences from industry, The Unified Modeling Language. *Modeling Languages, Concepts,and Tools*, vol. 2185, pp. 487–502.

Ani, Z.C. dan Basri,S., 2013. A Case Study of Effort Estimation in Agile Software Development Using Use Case Points. *Special Issue-Agile Symposium*, Malaysia, vol.25(4).

Bittner, K. & Spence, I., 2002. *Use Case Modeling.* Boston: Addison Wesley.

Clemmons,R.K.,2006. *Project Estimation With Use Case Points*," Diversified *Technical* Services, Inc.

Foss, T., Stensrud, E., Kitchenham, B., dan Myrtveit, I., 2002. *A Simulation Study of the Model Evaluation Citerion MMRE*. Discussion Paper 3/2002 , ISSN:0807 - 3406. Norwegian School of Management BI.

Hughes, B. dan Cotterell, M., 2006, *Software project management*, 4th Edition.

Institue, P. M., 2003. *Project Management Body of Knowledge.* Pennyslvania: Project Management Institue.

Karner, G. 1993. *Resource Estimation for Objectory Projects*. Objective Systems SF AB.

Koontz, H. dan O'Donnell, C., 1982. *Essentials of Management*. McGraw-Hill Inc., US.

Kurniawan, T. A. (2018). Pemodelan *Use Case* (UML): Evaluasi Terhadap beberapa Kesalahan dalam Praktik. *Jurnal teknologi Informasi dan Ilmu Komputer*, 5(1), 77. https://doi.org/10.25126/jtiik.201851610

Marchewka, J., 2003. *Information Technology Project Management*. Hoboken, NJ Wiley.

Nassif, A., Capretz, L.F., dan Ho, D., 2012. *Estimating Software Effort Using an ANN Model Based on Use Case Points. 11th International Conference on Machine Learning and Applications.*

Nassif, A., Capretz, L.F., dan Ho, D., 2011. *Regression Model for Software Effort Estimation Based on the Use Case Point Method*. *2011 International Conference on Computer and Software Modeling IPCSIT vol.14.*

Nassif, A., Capretz, L.F., dan Ho, D., 2010. Enhancing Use Case Points Estimation Method Using Soft Computing Techniques. *Journal of Global Research in*

Computer Science,[e-journal] 148(3).Tersedia di: <www.grcs.info> [Diakses 21Oktober2018]

Nassif, A., Capretz, L.F., dan Ho, D., 2012. *Software Effort Estimation in the Early Stage of The Software Life Cycle Using a Cascade Correlation Neural Network k Model. 13th International Conference on Software Engineering Artificial Intelligence.*

Ribu, K. 2001. Estimating Object-Oriented Software Projects with *Use case*s. *Master of Science Thesis*. University of Oslo Department of Informatics.

Saleh, K. 2011. *Effort and Cost Allocation in Medium to Large Software Development Projects*. International Journal of Computers (1), 74-79.

Schwalbe, K., 2004. *Information Technology Project Management*, 3th edition. Thompson, Canada.

Schwalbe, K., 2006. *Information Technology Project Management*, 4^{th} Ed., Thomson Course Technology

Schwalbe, K., 2012. *Information Technology Project Management*, 7th edition. Course Technology, Cengage Learning.

Schneider, G. Dan Winters, J., *1998. Applying Use Case – A Practical Guide. Addison-Wesley.*

Subriadi, A., Sholiq., dan Ningrum, P., 2014. *Critical Review Of The Effort Rate Value In Use Case Point Method for Estimating Software Development Effort. Journal of Theoretical and Applied Information Technology, vol.59(3)*.